
Noviembre 1985 · 250 Ptas.
TodoSpectrum
AÑO 2. - NUMERO 15.
REVISTA EXCLUSIVA PARA USUARIOS
QL MAGAZINE
Discos: más memoria a menos precio
nuevas utilidades: BCPL, Assembler, Monitor y Toolkit
Introduccion al SuperBasic
Introducción al Hardware
Puertas lógicas
Más potencia con el Beta Basic
una utilidad interesante
Basic letra a letra

SPECTRUM 128 EL SUMMUM

Spectrum, como líder, marca un nuevo hito en la historia de los ordenadores familiares.

El Spectrum 128.

Gran capacidad de memoria. Teclado y mensajes en castellano, teclado independiente para operaciones numéricas y de tratamiento de textos...

Sinclair e Investrónica han desarrollado una auténtica novedad. En ningún lugar del mundo, salvo en los Distribuidores Exclusivos de Investrónica, podrás encontrar el nuevo Spectrum 128.

Sé el primero en tener lo último.

SPECTRUM 128. NOVISIMUS



investronica

Tomás Bretón, 62.
Tel. (91) 467 82 10.
Telex 23399 IYCO E.
28045 Madrid

Camp, 80.
Tels. (93) 211 26 58 - 211 27 54.
08022 Barcelona

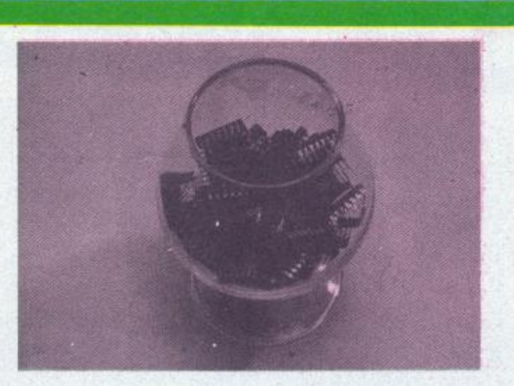


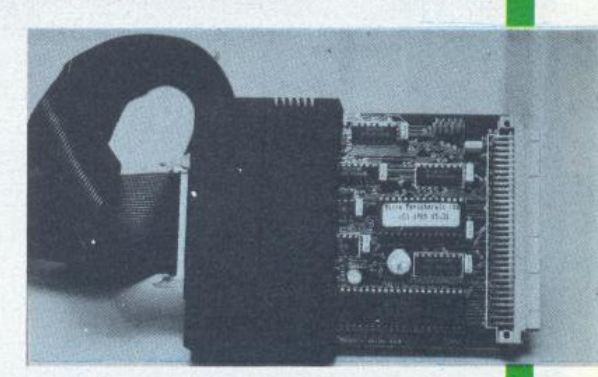
Portada: Puchecha

Director: Simeón Cruz

15

AÑO II









El tema del hardware parece ser el gran olvidado de todas las revistas de informática, que le dedican pocos artículos y ninguno de ellos que explique sus fundamentos. En este número intentamos remediar esta ausencia explicando qué es una puerta lógica.

Además incluimos una somera descripción de nuevo Spectrum 128 aunque posteriormente haremos un examen mucho más a fondo.

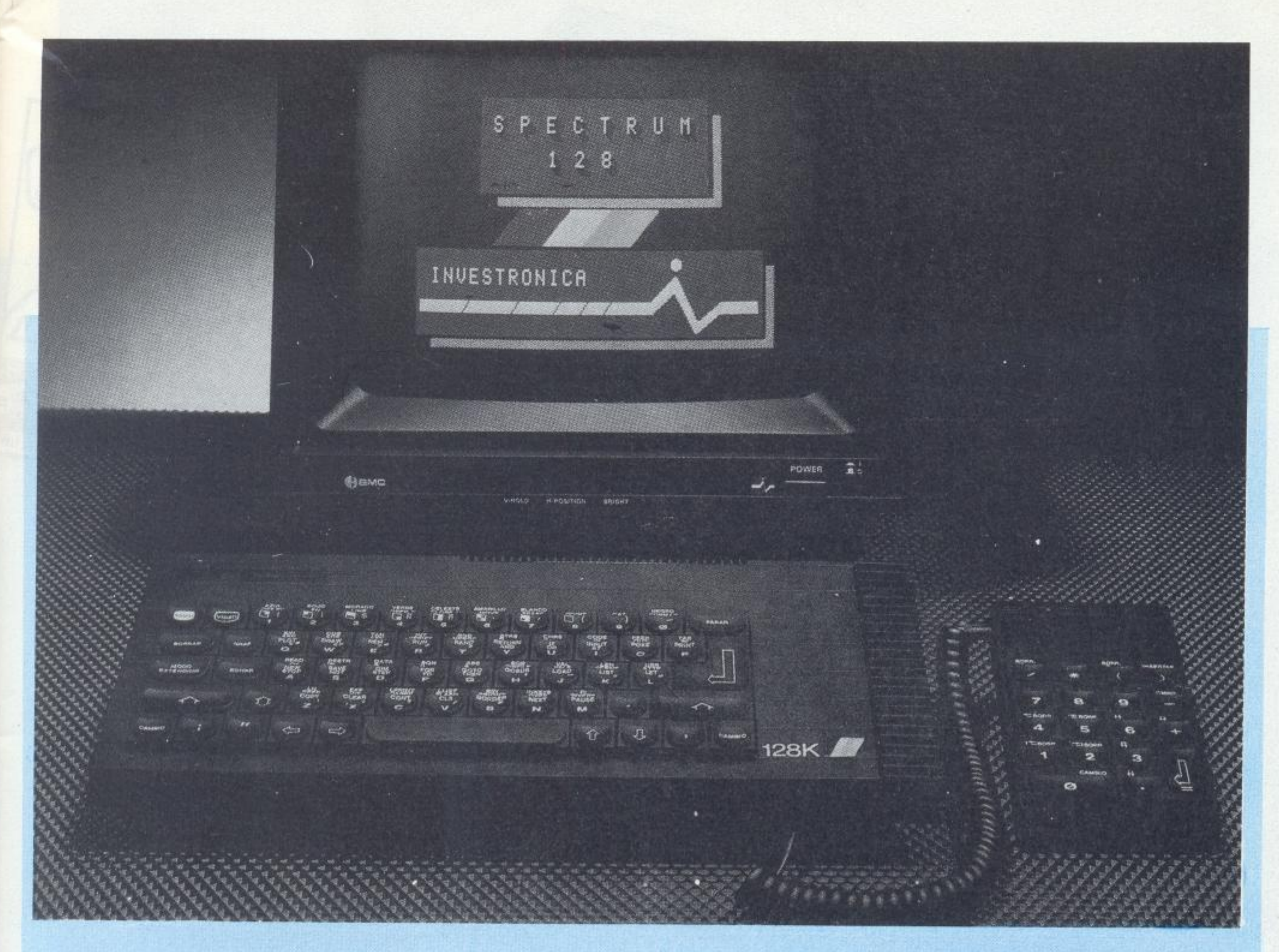
ACTUALIDAD

```
10 REM UN NUEVO SISTEMA DE PROTECCION. Para evitar la
       creciente pirateria, la casa inglesa ASAI ha inventado
       un sistema que muestra unos codigos especiales por
       pantalla y que deben ser tecleados para poder jugar.
       Lo interesante es que estos caracteres solo pueden
       verse con una lente especial que se incluye con el
       programa.

20 REM PROGRAMAS ESPAÑOLES PARA EL SPECTRUM 128. En el numero
       anterior comentabamos que diversas casas inglesas
       estaban trabajando en programas para este ordenador,
       pero las españolas tambien han puesto manos a la obra.
       Un ejemplo es ALSI Comercial que está adaptando todo
       su rango de programas comerciales (contabilidad,
       almacen, etc.).

30 REM SINCLAIR TIENE UN RESPIRO. Despues del fracasado
       acuerdo con Maxwell, Sinclair ha conseguido que sus
       acreedores esperen hasta despues de la campaña de
       navidad, en la que tienen puestas grandes esperanzas
       de venta.

40 REM JUEGO LOCO. La casa inglesa Silversoft acaba de
       presentar un nuevo juego de aventuras: "Bored of the
       Rings" (aburrido de los anillos), basado en un libro
       del mismo titulo que parodia el famoso "Señor de los
       Anillos", juego que a su vez ha hecho Melbourne House.
       Esperamos que ambos sean tan interesantes (y divertido
       el primero) como los libros a que hacen referencia.
```



Spectrum 128K.

La presentación en Barcelona durante el Sonimag del nuevo 128K, fue una sorpresa relativa ya que se esperaba desde hace tiempo una nueva máquina de Sinclair que viniera a colocarse como producto estrella dentro de la gama de ordenadores caseros de sir Clive, ya que el venerable Spectrum se encontraba agotado de llevar el liderazgo desde hace tiempo.

También hay que tener en cuenta que para una empresa que quiere seguir ostentando el liderazgo de este mercado venía siendo necesario sacar un producto que pudiese competir con los que la competencia iba a sacar. De este modo nos hemos encontrdo súbitamente con otras dos máquinas aparte de el Spectrum 128 con características similares en apariencia. El Amstrad CPC 6128 y el Commodore 128. Dado que las tres apuntan al mismo mercado, los entendidos señalan a que va a continuar la guerra que existía hasta ahora, pero con máquinas más potentes.

Para los actuales usuarios de Spectrum, o para aquellos que piensan compráserlo, resulta interesante saber que ofrece este nuevo producto y si las ampliaciones que le han hecho son realmente interesantes.

A primera vista

Debemos indicar, antes de empezar a hablar del ordenador, que la versión que vimos era un prototipo, lo que no nos permitió realizar un análisis en la profundidad que deseábamos, debiéndonos limitar a un examen general que nos permita determinar cómo será el producto final.

A primera vista se puede confundir el Spectrum 128 con el antiguo Spectrum+ (en realidad en el prototipo que examinamos ponía Spectrum+ además de 128), pero hay algunas características que lo diferencian. La primera y más llamativa es un teclado numérico independiente que va unido a la unidad central por medio de un cable de tipo teléfono. En este teclado además de las teclas de números propiamente dichas, existe otro ENTER, el punto, los paréntesis y las de las cuatro funciones elementales (suma, resta, división y multiplicación).

En el bloque central resalta la aparición de un disipador metálico de calor en la parte derecha. Muchos usuarios del Spectrum normal ya habían detectado el calor que se produce cuando la máquina lleva encendida bastante tiempo,





por lo que en este modelo y debido a la mayor cantidad de circuitos que lleva se hacía imprescindible esta medida que sin duda hará que la fuente de alimentación sufra menos.

Teclado

Por último hay que examinar el teclado en sí. La disposición de las teclas es exactamente igual que en el Spectrum+, medida que se habrá tomado para asegurar la compatibilidad seguramente, pero con los textos de algunas (sobre todo las que indican funciones) cambiados. De este modo "SYMBOL SHIFT" pasa a ser "CAMBIO", "EXTEND MODE" a "MODO EXTENDIDO", "DELETE" a "BORRAR" y "EDIT" a "EDITAR". Además los nombres de los colores pasan a castellano y TRUE VIDEO e INVERSE VIDEO han sido sustituidas por dibujos de pantallas de televisión con el fondo blanco y negro, respectivamente.

Aparte de estos cambios que podríamos denominar de estilo, el juego de caracteres también ha sido modificado para adaptarse a la norma de nuestro país. Por lo que se puede decir que ya existen eñes oficiales en el Spectrum. Pero debido al problema de la disposición del teclado, no están en su posición normal en una máquina de escribir, si no que la mayúscula está encima del cuatro y la minúscula encima del 5, sustituyendo a símbolo $ y al %, pasando estos a ocupar otras posiciones. Aparte de estos, hay que decir que la mayoría de los símbolos han sido cambiados de sitio y/o sustituidos por otros para introducir los correspondientes a nuestros idiomas, y decimos «nuestros», ya que se incluye la cedilla, usada en catalán. También están la «¿» y «¡», la «ü» y los acentos para ambos lados «´» y «`». Evidentemente los usuarios de los procesadores de textos amarán estas modificaciones.

Conexiones

En este modelo se ha cambiado la disposición de los conectores aunque básicamente permanencen los mismos aparte de algún añadido adicional. En la parte de atrás se encuentra el de alimentación, el de expasión (similar al del Spectrum y Spectrum+), el de televisor y un nuevo denominado RGB que permite conectarlo directamente a un monitor color cuando se desee calidad de imagen. Este conector es similar al del QL y por tanto un cable que valga para uno, valdrá también para el otro.

En lateral derecho se encuentran las dos conexiones de cassette, que han sido trasladadas aquí desde la parte trasera para facilitar la conexión y desconexión de cables, tema bastante engorroso en su antígua disposición. Al lado de estos se encuentra el conector RS 232. Este no existía en los modelos anteriores y su inclusión se debe a que es un estándar muy común y con múltiples aplicaciones; las comunicaciones entre ordenadores y la conexión a impresoras. En el tema de comunicaciones, en Inglaterra existe un mercado bastante desarrollado debido a la existencia de «Prestel», un sistema propiedad de la compañía telefónica que los usuarios pueden emplear mediante una subscripción. En él hay juegos, bases de datos con información de interés general, etc. Por desgracia nuestro país se encuentra muy atrasado en este aspecto y no existe nada parecido, por lo que la utilidad del RS 232 en este aspecto se ve bastante mermada.

Otro tema es la conexión de la impresora mediante este interfaz. Una cosa de la que el Spectrum normal estaba bastante necesitado era una conexión normalizada para que todo el software la pudiese usar sin problemas. Esto debería haberlo resuelto el interfaz 1, pero la realidad es que siguieron existiendo multitud de conexiones distintas, cada una con su propia versión. Al ir incorporando este interfaz en el nuevo modelo, la estandarización está asegurada y los usuarios se ahorrarán un montón de problemas, con el único requisito de que la impresora se comunique en serie en vez de en paralelo como venía siendo común. En este mismo conector se encuentra el interfaz MIDI. Todos aquellos aficionados a la música habrán oído hablar alguna vez de él, ya que se trata de una norma existente que permite que los instrumentos musicales electrónicos que también lo tengan puedan ser gobernados por el ordenador. De este modo se pueden generar hasta ocho notas simultáneas en cualquier instru-



mento de tipo profesional (órganos, sintetizadores, baterías electrónicas, etc.) si se es aficionado a la música y se desea tener todo un estudio de grabación, pero no encarece excesivamente el precio a los que no lo necesiten.

Por último hay que indicar la existencia de un enchufe para el teclado numérico en la parte delantera y el habitual botón de RESET a la izquierda de la máquina.

Interioridades

Pero donde las novedades son más importantes es en el interior de la máquina. La placa principal ha sido rediseñada para dar cabida a los nuevos componentes. En primer lugar la memoria RAM se ha ampliado de 48K de modelos anteriores a 128K. Esto se ha conseguido poniendo dos bloques de 64K que son manejados por el microprocesador y alguna circuitería lógica lo que permite su uso conjunto por medio de lo que se denomina «mapeado de memoria» o «paginado». Con este sistema se puede manejar todo para introducir programas en código máquina o desde BASIC, aunque en este caso con algunas limitaciones como señalamos en el apartado de software.

La ROM también ha sido ampliada considerablemente para hacer sitio a los nuevos comandos que permiten el manejo de la memoria extra, así como para incluir las instrucciones de modelos anteriores que permiten la compatibilidad. Por otro lado se encuentra la circuitería encargada del manejo del puerto serie y del interface MIDI, que como es habitual en los diseños de Sinclair son controladas directamente por el Z-80.

Un último detalle lo constituye el conector trasero, de la misma forma y con la misma distribución de señales que en el Spectrum, lo que permite que los periféricos ya existentes sean usados por este modelo cuando funciona en modo simulación de 48K, aunque en el modo normal pueden existir algunos problemas debidos al mapeado de memoria y a la ROM distinta.

Software

El ordenador tiene dos modos de funcionamiento distintos. Según se enciende actúa con la nueva ROM, lo que provoca bastantes cambios con respecto a lo habitual. En primer lugar las órdenes se teclean letra a letra en lugar de generarse con una sola pulsación. Esto hará felices a muchas personas que prefieren ese sistema, que aunque teóricamente más largo, es más fácil de usar. Por otro lado existe un conjunto más amplio de instrucciones que permiten el manejo de la RAM extra como si fuese un disco (o microdrive) en RAM. Esto ayuda en el manejo de gran cantidad de datos pero imposibilita su uso para la escritura de programas muy largos debido a que el intérprete de BASIC sigue creyendo que el espacio para programas es el habitual. Esto en realidad no es tan grave, ya que es bastante extraña la existencia de programas tan largos, siendo mucho más común un programa de dimensiones más reducidas pero que maneje gran cantidad de datos. En especial las bases de datos que salgan serán mucho más potentes que las existentes actualmente. Un detalle a destacar es que los mensajes de error han sido traducidos al castellano lo que permitirá que la rabia que entra cuando un programa nos da un error sea mayor aún, al saber cuál es el fallo.

El milagro de la compatibilidad se consigue tecleando "SPECTRUM", convirtiéndose la máquina en un Spectrum+, lo que permite la ejecución de gran cantidad de programas comerciales existentes. Normalmente no habrá problemas con éstos, aunque puede presentarse el caso de programas muy específicos que utilicen determinadas llamadas a la ROM y que generen problemas, aunque este caso no es muy común, por lo que conviene ir acostumbrándose a que a partir de ahora las cintas comerciales salgan con una pegatina que diga «Compatible Spectrum 128».

Conclusión

La máquina ofrece unas prestaciones prometedoras a un precio que promete ser muy apetitoso, colocándose con ventaja sobre sus competidores directos, sobre todo considerando la base de software existente y la que vendrá, ya que no hay que dudar que todas las casas de software importante están trabajando ya en ello. En cualquier caso; ¡Bienvenido Spectrum 128!

Circulos redondos

El comando CIRCLE es una de las funciones no estándar del ZX-BASIC. Quienes lo hayan visto en funcionamiento no tendrán muchos problemas en admitir que probablemente hay métodos para hacer las circunferencias más rápidamente.

En este artículo proponemos un programa que es capaz de realizar círculos a una considerable velocidad. Y no sólo esto. Las circunferencias podrán ser parcialmente visibles. Además, aprenderemos interesantísimas propiedades de los DEF FN, que nos van a permitir pasar cómodamente datos a un programa en código máquina.

¿Por qué tarda tanto el Spectrum en hacer un círculo? La razón es que el programa residente en ROM encargado de gestionar el comando CIRCLE es innecesariamente complicado. Para su funcionamiento se hace uso de aritmética en punto flotante —y ya sabemos todos que eso no es el fuerte del Spectrum—. Al principio del programa se usan las funciones trigonométricas, de reconocida lentitud. También son necesarias multiplicaciones, que alargan el tiempo de ejecución del proceso. El programa 1 muestra de qué manera se ha llevado a cabo el programa ROM. El dibujo se realiza a partir de la unión de segmentos.

Calcularemos en primer lugar el ángulo que produce tramos rectos, y a continuación, entramos en un bucle que calcula las nuevas coordenadas para, seguidamente, unir los puntos.

En nuestra cómoda misión de críticos implacables podemos poner alguna pega al método. Ya que se ha decidido usar el procedimiento expuesto, no hubiera costado nada añadir un cuarto parámetro "e" —excentricidad— que nos permitiría dibujar elipses. Si damos el valor 1, obtendremos las viejas circunferencias. Valores más pequeños ofrecerán elipses "horizontales", y más grandes, verticales. Para verlo, realicemos las siguientes modificaciones en nuestro programa:

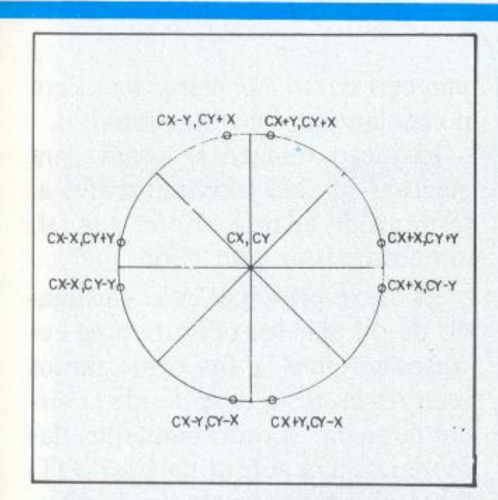
```
10 INPUT "CIRCLE"; "CX;", ";CY;", ";R;", ";E
90 DRAW -X, E*Y
```

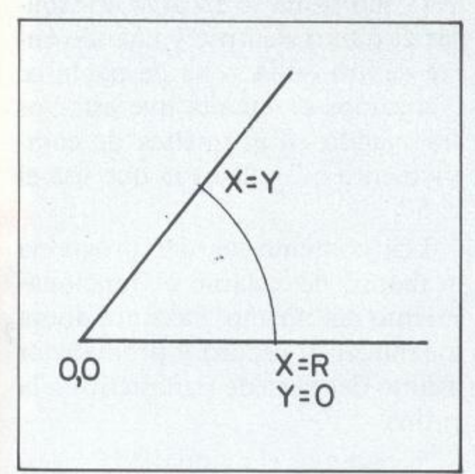
Y si hemos decidido no dibujar elipses, podemos sacar un gran partido de la simetría de la circunferencia. La figura 1 nos ayuda a ver cómo basta con calcular únicamente un sector de 45 grados. El resto de la figura puede determinarse fácilmente a partir de estos valores. De esta manera puede reducirse enormemente el tiempo de cálculo y por tanto, el total.

Veamos la figura 2, donde se dibuja un octavo de círculo centrado el 0,0. Si empezamos desde abajo los valores iniciales de las coordenadas serán: X = Radio, Y = O. Mirando con detalle a la circunferencia ya digitalizada, vemos que, partiendo de este punto, hay sólo dos posibles modos de evolución:

i) X2 = X1
   Y2 = Y1+1
ii) X2 = X1−1

Y2 = Y1+1, esto es, o movernos verticalmente hacia arriba, o dia-





gonalmente hacia el vértice superior-izquierdo.

Si nosotros supiéramos en qué momento tenemos que tomar alguna de las direcciones, habríamos solucionado nuestro problema de una forma muy elegante.

Métodos hay muchos, y como siempre, unos mejores y otros peores. El escogido se debe a **J. Michener,** que se basó en un algoritmo de **J. Breseham,** desarrollado inicialmente para un *plotter.* Presenta varios puntos de especial interés, siendo el más importante que usa aritmética de enteros, que puede

```
 10  REM Programa 1
 20  REM
 30  INPUT " CIRCLE ";CX;",";CY
     ;",";R
 40  LET A=ACS (1-1/R)
 50  LET C=2*PI /A
 60  LET SA=SIN A: LET CA=COS A
 70  LET X=0: LET Y=R*SA
 80  PLOT CX+R,CY-Y/2
 90  DRAW -X,Y
100  LET C=C-1: IF INT C=0 THEN
      STOP
110  LET TMPX=X
120  LET X=Y*SA+X*CA
130  LET Y=Y*CA-TMPX*SA
140  GO TO 90
```

```
  1  REM Programa 2
  5  REM
 10  INPUT " CIRCLE ";CX;",";CY
     ;",";R
 20  LET E=3-2*R: LET X=R: LET
     Y=0
 30  IF Y>X THEN  STOP
 40  PLOT CX+X,CY+Y
 50  PLOT CX-X,CY+Y
 60  PLOT CX+X,CY-Y
 70  PLOT CX-X,CY-Y
 80  PLOT CX+Y,CY+X
 90  PLOT CX-Y,CY+X
100  PLOT CX+Y,CY-X
110  PLOT CX-Y,CY-X
120  IF E<0 THEN LET E=E+4*Y+6
     : GO TO 140
130  IF E>=0 THEN LET X=X-1: L
     ET E=E+(Y-X)*4+10
140  LET Y=Y+1: GO TO 30
```

implementarse a base de sumas únicamente. Y dibuja círculos con un aspecto excelente.

El programa 2 ilustra el principio de funcionamiento de la rutina. Existirá un término de error E que será el que nos indique con su signo si debemos realizar una maniobra u otra. En cada caso, el valor siguiente que tome será diferente.

Las líneas 40 a 110 hacen posible el dibujo de la circunferencia completa a partir de un único sector.

El programa BASIC ya es rápido, pero no es nada comparado con la versión código máquina. Si no lo creen, echen un vistazo a la tabla que hemos confeccionado.

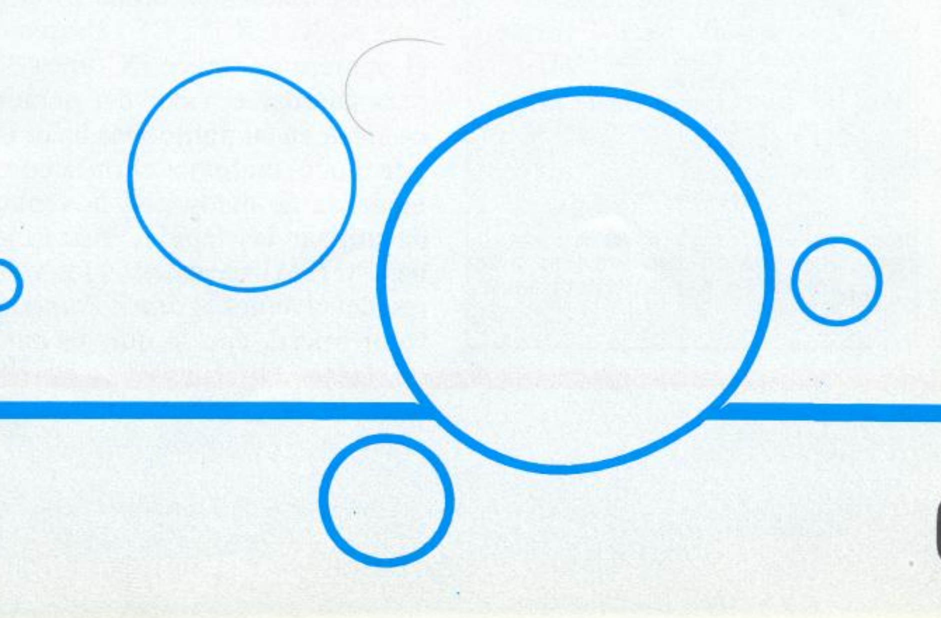
Tiempo empleado para dibujar 1000 circunferencias (segundos)

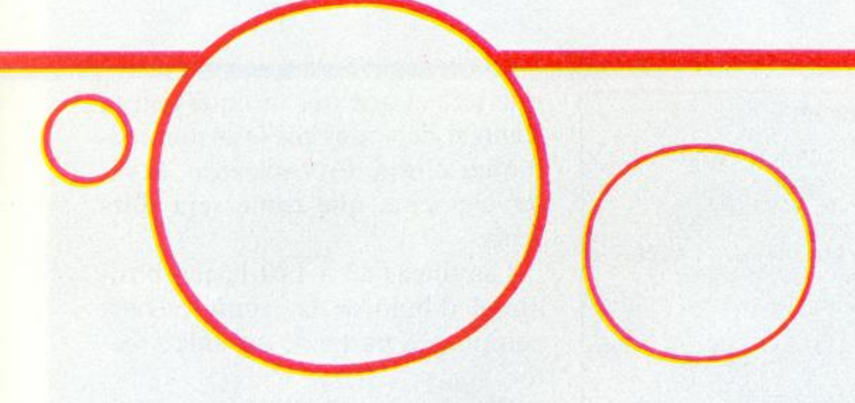
RADIO:	10	40	70	100
Rutina	20	45	71	80
CIRCLE	490	780	920	–

Invitamos a los curiosos a demostrar que existe una relación de proporcionalidad entre el radio y el tiempo.

Pasamos ya al programa. Su funcionamiento es muy sencillo, por lo que únicamente perfilaremos su modo de actuar.

Antes de nada, es importante hacer notar que, dado que vamos a usar el juego de registros imagen (el famoso *alternate register set*) será interesante (más que intere-





```
10  DATA 217,229,243,221,42,11
    ,92,221,78,4,221,70,5,221,9
    4,12
20  DATA 221,86,13,221,110,20,
    221,102,21,213,17,0,0,213,1
    97,229
30  DATA 221,33,0,0,221,57,229
    ,217,225,41,235,33,3,0,167,
    237
40  DATA 82,217,167,237,82,250
    ,166,91,62,2,8,209,225,25,2
    35,193
50  DATA 225,9,221,249,205,175
    ,91,209,225,25,235,193,225,
    167,237,66
60  DATA 221,249,205,175,91,20
    9,225,167,237,82,235,193,22
    5,9,221,249
70  DATA 205,175,91,209,225,16
    7,237,82,235,193,225,167,23
    7,66,221,249
80  DATA 205,175,91,193,225,20
    9,197,229,213,8,61,194,58,9
    1,197,217
90  DATA 235,203,122,40,8,225,
    41,41,1,6,0,24,12,225,193,1
    1
100 DATA 197,167,237,66,41,41,
    1,10,0,9,25,217,225,209,209
    ,19
110 DATA 213,221,249,195,50,91
    ,33,8,0,57,249,225,217,251,
    201,122
120 DATA 167,192,124,167,192,6
    2,175,189,216,69,75,205,170
    ,34,71,4
130 DATA 62,1,15,16,253,182,11
    9,201
200 LET SUM=0: FOR I=0 TO 199:
     READ A: POKE (23296+I),A:
    LET SUM=SUM+A: NEXT I
210 IF SUM<>27735 THEN  BEEP .
    2,10: PRINT "Error en el DA
    TA": STOP
300 REM
310 REM   Demostracion
320 REM
330 DEF FN C(X,Y,R)=USR 23296
340 FOR I=0 TO 180 STEP 5: RAN
    DOMIZE FN C(128,88,I): NEXT
    I
```

sante, necesario), guardar el valor de HL' en el *stack*. Antes de retornar al BASIC devolveremos los antiguos valores. En efecto, el intérprete necesita la información de este registro para volver al lugar de origen.

Dejaremos para el final el comentario sobre el paso de parámetros al programa a partir del DEF FN. Su especial interés así lo aconseja.

De este modo nos presentamos en la línea 23 guardando en HL el valor del radio, en BC la coordenada X del centro y en DE la coordenada Y.

El funcionamiento del programa circula en torno al *stack*. Guardaremos por este orden, las coordenadas Y del centro y la asociada al punto (lo que hemos llamado Y en el programa BASIC). A continuación, lo propio en X. A la hora de sacar los datos del *stack*, obtendremos los valores en orden inverso, esto es, X, CX, Y, CY. Usaremos el puntero de índice IX (línea 30) para guardar el valor del puntero de *stack* en su punto más bajo. De este modo, usaremos el *stack* como memoria de datos, con la ventaja de utilizar las rápidas instrucciones POP. Al sacar todos los valores, devolvemos al *Stack Pointer* el valor inicial, con lo que los datos parecen como no recogidos. Pero no adelantemos acontecimientos.

El juego imagen se usará para efectuar los cálculos referentes al término de error E. En el par HL almacenaremos este valor.

El bucle principal es el encargado de dibujar los ocho puntos correspondientes a los otros tantos sectores en los que se divide la circunferencia. Continuamente llamaremos a la subrutina CLPLOT. Esta subrutina se encarga de dibujar el punto siempre y cuando entre dentro de la zona de pantalla. Tengamos en cuenta que estamos trabajando en aritmética de complemento a 2 que es la que usa el Z-80.

Los comentarios del programa acabarán de aclarar el funcionamiento del mismo. Pasamos ahora a explicar el oscuro y prometedor asunto del paso de parámetros a la rutina.

Si cogemos el manual de Programación (nunca nos cansaremos de recomendarlo) y lo abrimos en el capítulo 25, nos encontramos con que existe una variable del sistema conocida como DEFADD, que guarda la dirección del argumento de una función definida por el usuario. Esto nos debería llevar a investigar un poco en este sentido. Las conclusiones sacadas son muy interesantes. Resulta que después de la definición de cada una de las variables de la función hay espacio para almacenar los valores de los parámetros que pasamos a ésta. Es la única vez en la que se trabaja directamente sobre el área de programa, operando sobre ella. Veamos

un caso concreto para fijar ideas:

DEF FN C (U, V, W)=...

Cuando se localiza un FN, se busca la definición, y se procede a transferir los valores de la línea de llamada a la del DEF FN. A continuación, pasaremos a trabajar en esta línea. Cuando aparece una variable, husmearemos primero en nuestro particular área de variables. En caso de no encontrarlo, pasaríamos entonces a buscarlo entre las auténticas variables.

Si resulta que la definición de la función es un USR, nuestra rutina recibirá el control del ordenador cuando DEFADD apunta a la dirección del primer parámetro del DEF FN, concretamente al nombre de la variable. Los parámetros deben ser necesariamente de una letra, por lo que sabremos inmediatamente dónde empieza el espacio dedicado al número. Y si el valor asignado es un entero comprendido entre –65535 y +65535 (ver capítulo 24 del Manual), los números se almacenarán en un formato especial (no el de coma flotante), por lo que encontraremos en los *bytes* tercero y cuarto de los dedicados al número, nuestro valor en el formato que usa el Z-80. Incluso conservaremos el signo.

Vamos a expresar gráficamente la asignación de valores en el programa ejemplo.

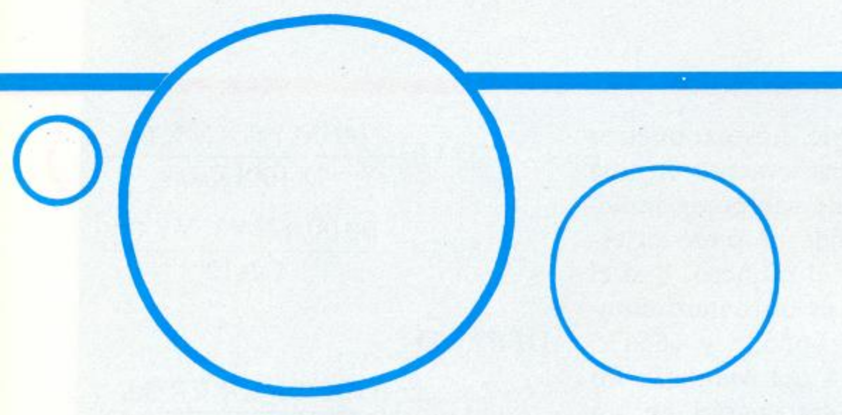
58 OF $\dfrac{\text{00 00/FF XX X 00}}{\text{IX+00IX+04}}$

2C 59 OF $\dfrac{\text{00 00/FF YY YY 00}}{\text{IX+12}}$

DEFADD

...52 OF $\dfrac{\text{00 00/FF RR RR 00}}{\text{IX+20}}$

El programa 4 permite introducir la rutina en el *buffer* de impresora. Una última recomendación: recuerde que siempre que hagamos uso del FN deberemos usar números enteros como parámetros. Si no lo hiciéramos así, puede pasar un buen rato antes de que recuperemos el control del aparato.



```
;   ******************
;   ****  CIRCLE  ****
;   ******************


        ORG  23296

        EXX
        PUSH HL

        DI

        LD   IX,(5C0BH);DEF_ADD
        LD   C,(IX+4) ;Coord X
        LD   B,(IX+5)
        LD   E,(IX+12);Coord Y
        LD   D,(IX+13)
        LD   L,(IX+20);Radio
        LD   H,(IX+21)


        PUSH DE;        CY
        LD   DE,0000
        PUSH DE;        Y : 0
        PUSH BC;        CX
        PUSH HL;        X : R

        LD   IX,0000
        ADD  IX,SP;Apuntamos a la
;            tabla

;Primer valor de E
        PUSH HL
        EXX
        POP  HL
        ADD  HL,HL; 2*RADIO
        EX   DE,HL
        LD   HL,0003
        AND  A
        SBC  HL,DE
        EXX  ;     E = 3 - 2 * RAD

LOOP    AND  A
        SBC  HL,DE; IF Y>X THEN
        JP   M,EXIT;   se acabo

        LD   A,2;Contador  bucle
LPI     EX   AF,AF'
;El bucle a continuacion calcula
;y dibuja las coordenadas de ca-
;da sector.

;   1a pasada      2a pasada
;   CX+X,CY+Y      CX+Y,CY+X
;   CX+X,CY-Y      CX+Y,CY-X
;   CX-X,CY+Y      CX-Y,CX+X
;   CX-X,CY-Y      CX-Y,CX-X

        POP  DE; X(1a vez), Y(2a)
        POP  HL; CX
        ADD  HL,DE
        EX   DE,HL
        POP  BC; Y(1a vez), X(2a)
        POP  HL; CY
        ADD  HL,BC
        LD   SP,IX; Dejamos SP
;            como estaba.
        CALL CLPLOT

        POP  DE; X(1a), Y(2a)
        POP  HL; CX
        ADD  HL,DE
        EX   DE,HL
        POP  BC; Y(1a), X(2a)
        POP  HL; CY
        AND  A
        SBC  HL,BC
        LD   SP,IX
        CALL CLPLOT

        POP  DE; X(1a), Y(2a)
        POP  HL; CX
        AND  A
        SBC  HL,DE
        EX   DE,HL
        POP  BC; Y(1a), X(2a)
        POP  HL; CY
        ADD  HL,BC
        LD   SP,IX
        CALL CLPLOT

        POP  DE; X(1a), Y(2a)
        FOP  HL; CX
        AND  A
        SBC  HL,DE
        EX   DE,HL
        POP  BC; Y(1a), X(2a)
        POP  HL; CY
        AND  A
        SBC  HL,BC
        LD   SP,IX
        CALL CLPLOT

        POP  BC; Intercambiamos
        POP  HL
        POP  DE; X e Y en el
        PUSH BC
        PUSH HL; stack.
        PUSH DE

        EX   AF,AF'
        DEC  A
        JP   NZ,LPI
        PUSH BC; Guardamos Y

; Actualizamos el termino de
;error E
        EXX
        EX   DE,HL
        BIT  7,D; Signo de E
        JR   Z,POSIT

;IF E<0 THEN...
        POP  HL;     Y
        ADD  HL,HL; 2*Y
        ADD  HL,HL; 4*Y
        LD   BC,0006
        JR   J1

;IF E>=0 THEN...
POSIT   POP  HL;     Y
        POP  BC;     X
        DEC  BC;     X=X-1
        PUSH BC;Nueva coordenada
        AND  A
        SBC  HL,BC; Y-X
        ADD  HL,HL
        ADD  HL,HL; 4*(Y-X)
        LD   BC,0010
J1      ADD  HL,BC
        ADD  HL,DE;E=E+4*Y+4 o
;                 E=E+4*(Y-X)+10

; Incrementamos coordenada X
        EXX
        POP  HL;     X
        POP  DE
        POP  DE;     Y
        INC  DE;     Y=Y+1
        PUSH DE
        LD   SP,IX;Dejamos el SP
;            como estaba.
        JP   LOOP; Otra vez.

EXIT    LD   HL,0008
        ADD  HL,SP; Vaciamos PILA
        LD   SP,HL

;            Preparados para
        POP  HL; volver al BASIC
        EXX
        EI
        RET

;Subrutina CLPLOT
CLPLOT  LD   A,D
        AND  A
        RET  NZ;Solo X>0 y X<256

        LD   A,H
        AND  A
        RET  NZ;Solo Y>0 e Y<256

        LD   A,175
        CP   L
        RET  C; Solo Y<175

        LD   B,L; Coord X
        LD   C,E; Coord Y

        CALL 22AAH;PIXEL ADDR
;       Proporciona en HL la dir
;       del pixel y en A el bit

        LD   B,A
        INC  B
        LD   A,01
ROT     RRCA
        DJNZ ROT

        OR   (HL)
        LD   (HL),A
        RET
```

SERVICIO DE EJEMPLARES ATRASADOS

Complete su colección de

Todospectrum

A continuación le resumimos el contenido de los ejemplares aparecidos hasta ahora.

Núm. 1 • 250 pts.
Cómo usar el microdrive/Programación Basic/Ampliación Basicare/Rutina despertador/Variables del sistema/Entrada datos mediante máscaras/Protección del software/Sintonice su Spectrum/Programas.

Núm. 2 • 250 pts.
Gráficos profesionales/Desplazamiento pixel a pixel/Utilización de rutinas/Construcción del interface centronics/Programas de utilidad para microdrive/Rutina reset en código máquina/Análisis del editor de textos Tasword/Interfaces para impresoras/Programas.

Núm. 3 • 250 pts.
Novedades sonimag '84/Ampliando el Basic/Programas para ordenar programas/Gráficos con el VU-3D/Lenguaje Forth/Archivos en microdrive/Programación de un interface de impresora/Programas.

Núm. 4 • 250 pts.
De profesión: programador/Consola para el Spectrum/Comparación código máquina-Basic/Análisis programa contabilidad /Calendario/Pascal/Programas.

Núm. 5 • 250 pts.
Floppys para Spectrum/Diseño asistido por ordenador/64 Caracteres por línea/Juego de la vida/Pascal/Así hacemos las portadas/Control de evaluaciones/Programas.

Núm. 6 • 250 pts.
Representación de funciones/Todos los caminos conducen a la ROM/Juegos/Pascal/Construcción de un lápiz óptico/Programas de gestión. El SITI/Logo: tortugas para todos/Interrupciones del Z-80/Programas.

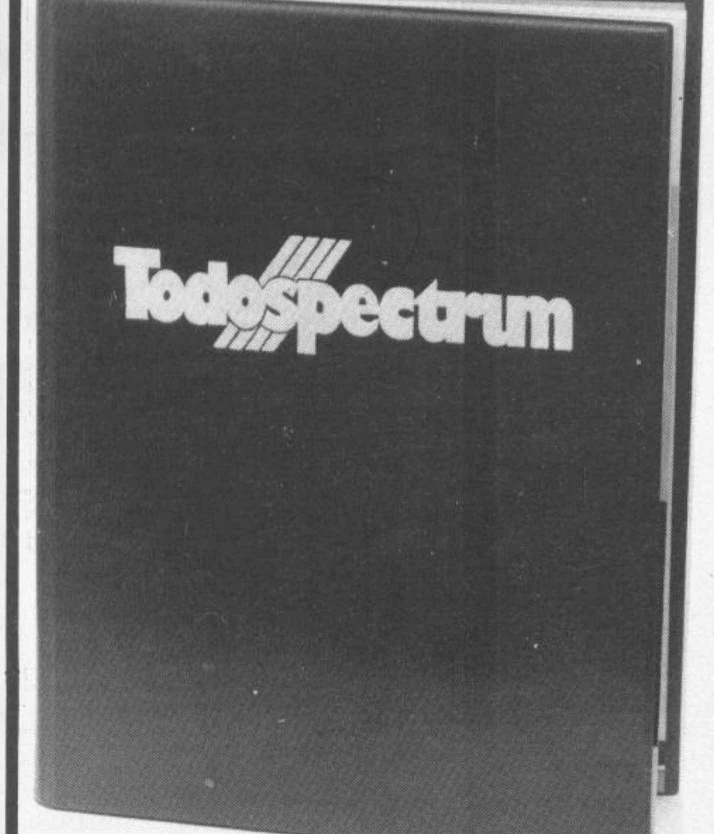




DISPONEMOS DE TAPAS ESPECIALES PARA SUS EJEMPLARES DE

Todospectrum

SIN NECESIDAD DE ENCUADERNACION



(cada tapa es para 6 ejemplares)



Para hacer su pedido, rellene este cupón HOY MISMO y envíelo a:

Todospectrum
Bravo Murillo, 377
Tel. 733 96 62 - 28020 MADRID

Ruego me envíen los siguientes ejemplares atrasados de TODOSPECTRUM .... al precio de 250 pts.

Por favor envíenme .... tapas para la encuadernación de mis ejemplares de TODOSPECTRUM, al precio de 600 pts. más gastos de envío.

El importe lo abonaré
☐ POR CHEQUE ☐ CONTRA REEMBOLSO ☐ CON MI TARJETA DE CREDITO ☐ AMERICAN EXPRESS ☐ VISA ☐ INTERBANK

Número de mi tarjeta: ☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐☐

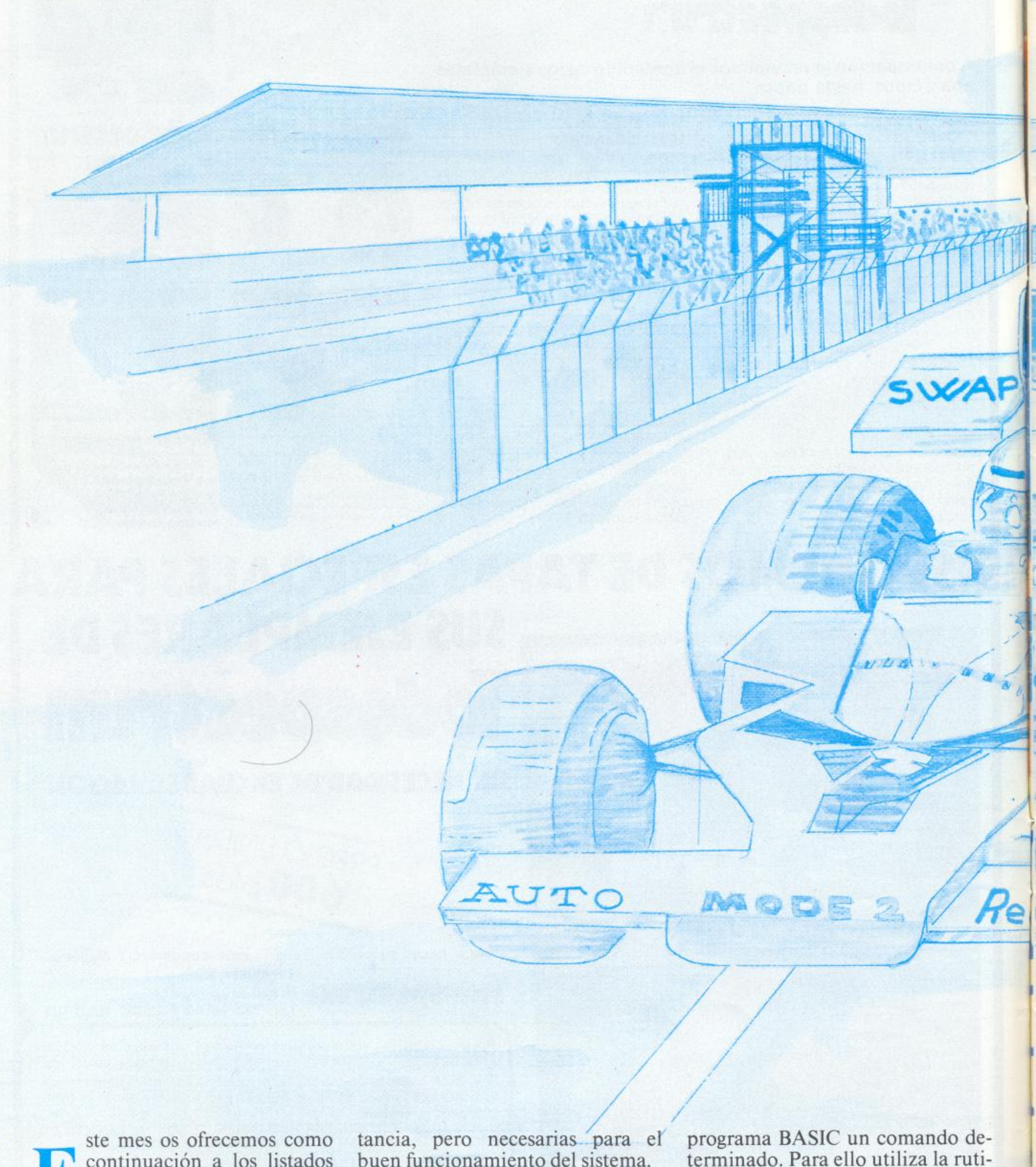
Fecha de caducidad .... Firma

NOMBRE ....

DIRECCION ....

CIUDAD .... C. P. ....

PROVINCIA ....

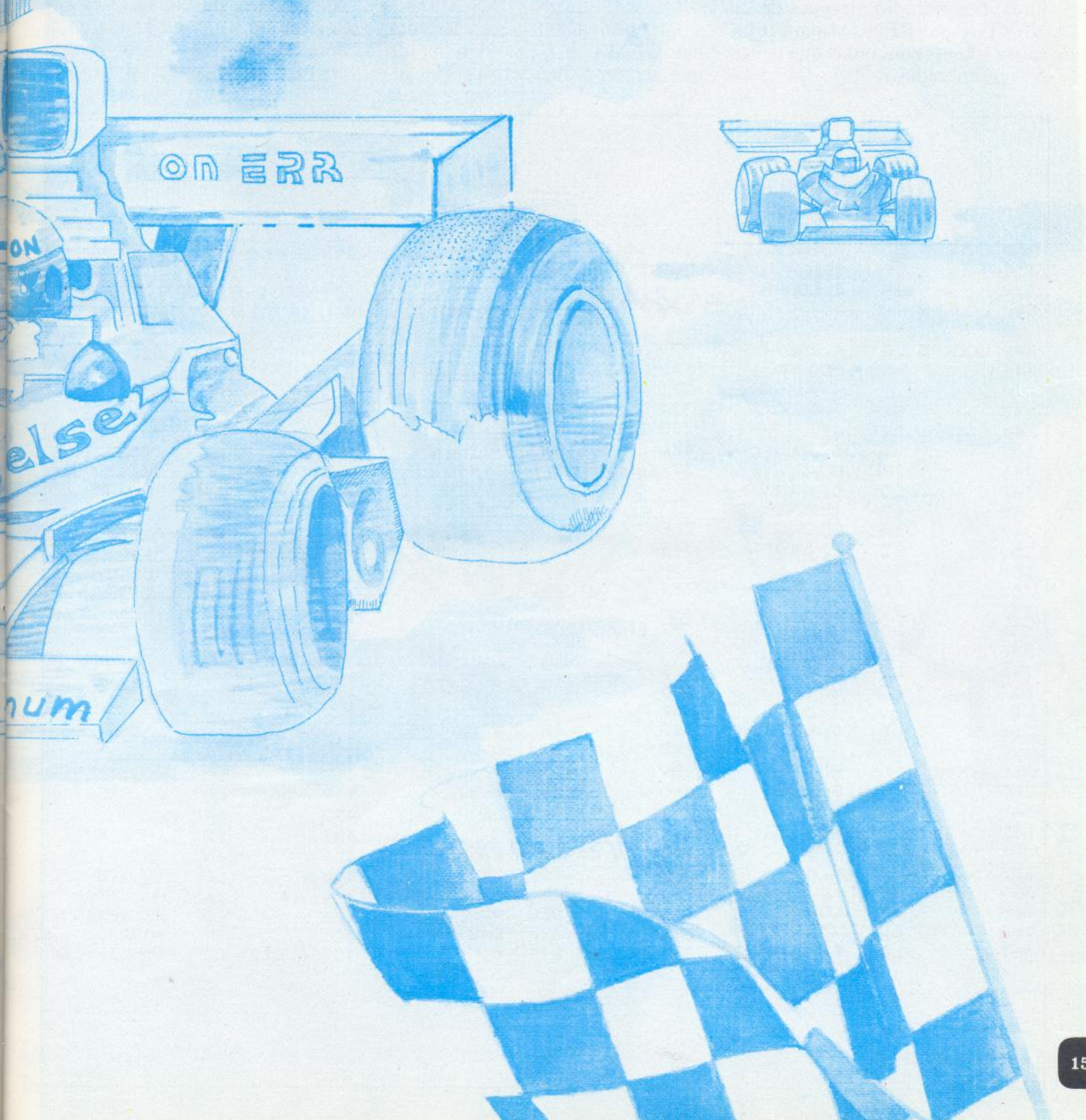


Este mes os ofrecemos como continuación a los listados comentados del programa, la parte correspondiente al editor y al intérprete BASIC. También están intercaladas algunas rutinas de menor importancia, pero necesarias para el buen funcionamiento del sistema.

Diversas rutinas

La primera rutina es LOOKPR. Su función es buscar a lo largo del programa BASIC un comando determinado. Para ello utiliza la rutina EASTMT, que rastrea una línea en busca del comando. LOOKPR es utilizada cuando se produce un error para buscar el comando ONERR. Se utiliza con

Un nuevo operativo para el Spectrum ③

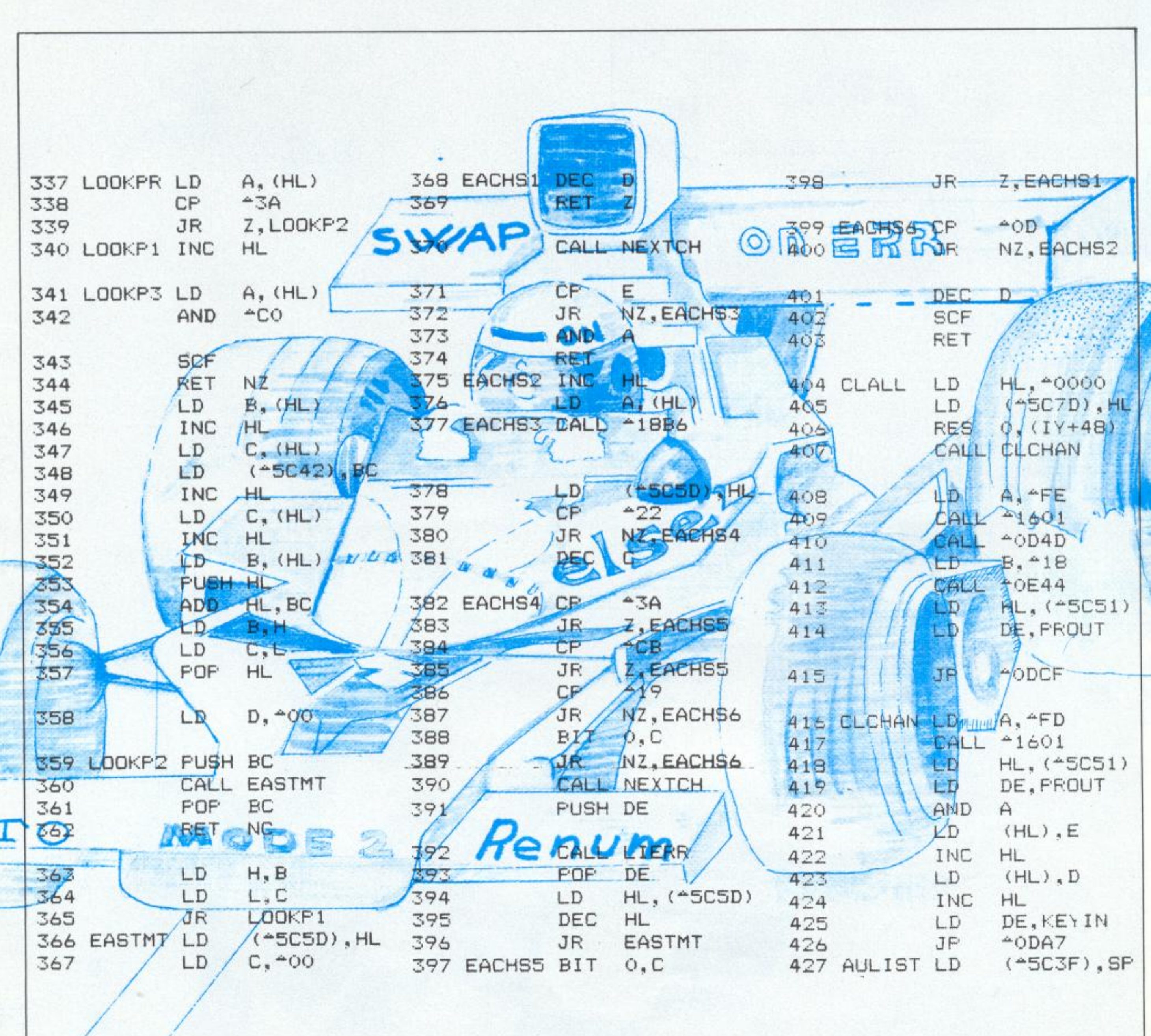


dos puntos de entrada, LOOKP3 para comenzar la búsqueda y LOOKP2 para continuarla. EASTMT, aparte de por LOOKPR, también es usada por NEXTLI para localizar un número de sentencia concreto cuando se produce un salto al interior de una línea, por ejemplo, después de un NEXT, y por RENUM para ir localizando los comandos que deben ser renumerados.

La función de OUTLI2 es copiar una línea del programa en la pantalla.

A continuación vienen las rutinas CLALL y CLS. Han tenido que ser reproducidas por que incluyen a CLCHAN, que hace que el canal «K» utilice como direcciones de salida y entrada las normales de la ROM, lo cual anularía los dos modos de presentación y la lectura de teclado. También cambia la dirección de salida del canal «S».

La función de OUTLI2 es copiar una línea del programa en la pantalla. El cursor se escribe en inverso en un lugar de en FLASH




```
LOOKPR LD   A,(HL)
       CP   ^3A
       JR   Z,LOOKP2
LOOKP1 INC  HL

LOOKP3 LD   A,(HL)
       AND  ^C0

       SCF
       RET  NZ
       LD   B,(HL)
       INC  HL
       LD   C,(HL)
       LD   (^5C42),BC
       INC  HL
       LD   C,(HL)
       INC  HL
       LD   B,(HL)
       PUSH HL
       ADD  HL,BC
       LD   B,H
       LD   C,L
       POP  HL

       LD   D,^00

LOOKP2 PUSH BC
       CALL EASTMT
       POP  BC
       RET  NC

       LD   H,B
       LD   L,C
       JR   LOOKP1
EASTMT LD   (^5C5D),HL
       LD   C,^00
EACHS1 DEC  D
       RET  Z

       CALL NEXTCH

       CP   E
       JR   NZ,EACHS3
       AND  A
       RET
EACHS2 INC  HL
       LD   A,(HL)
EACHS3 CALL ^18B6

       LD   (^5C5D),HL
       CP   ^22
       JR   NZ,EACHS4
       DEC  C

EACHS4 CP   ^3A
       JR   Z,EACHS5
       CP   ^CB
       JR   Z,EACHS5
       CP   ^19
       JR   NZ,EACHS6
       BIT  0,C
       JR   NZ,EACHS6
       CALL NEXTCH
       PUSH DE

       CALL LIERR
       POP  DE
       LD   HL,(^5C5D)
       DEC  HL
       JR   EASTMT
EACHS5 BIT  0,C
       JR   Z,EACHS1

EACHS6 CP   ^0D
       JR   NZ,EACHS2

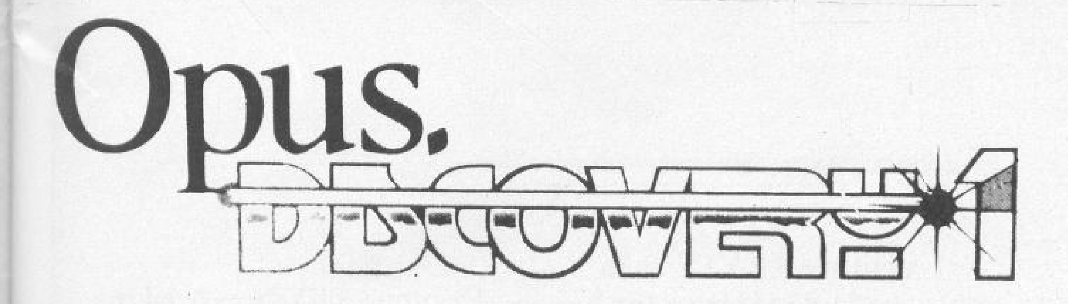
       DEC  D
       SCF
       RET

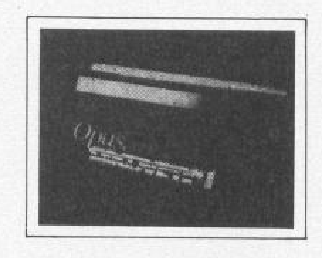
CLALL  LD   HL,^0000
       LD   (^5C7D),HL
       RES  0,(IY+48)
       CALL CLCHAN

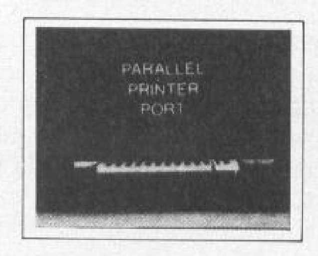
       LD   A,^FE
       CALL ^1601
       CALL ^0D4D
       LD   B,^18
       CALL ^0E44
       LD   HL,(^5C51)
       LD   DE,PROUT

       JP   ^0DCF

CLCHAN LD   A,^FD
       CALL ^1601
       LD   HL,(^5C51)
       LD   DE,PROUT
       AND  A
       LD   (HL),E
       INC  HL
       LD   (HL),D
       INC  HL
       LD   DE,KEYIN
       JP   ^0DA7
AULIST LD   (^5C3F),SP
```







UN SISTEMA COMPACTO DE DISCO PARA EL SPECTRUM Y EL SPECTRUM +



El Discovery 1 es el más avanzado sistema de disco jamás desarrollado para el Spectrum.

Diseñado en Inglaterra, incorpora la más reciente tecnología japonesa en discos de 3,5". El Discovery es el primero de la nueva generación de periféricos para el Spectrum.

Las asombrosas características del Discovery incluyen:

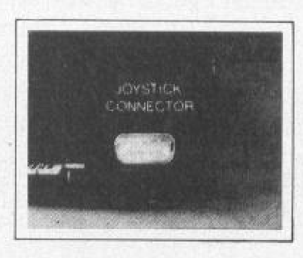
- El más moderno sistema de disco de 3,5"
- Salida Centronics para impresora.
- Interface de Joystick tipo Kempston
- Salida para monitor monocromo.
- Alimentación propia y para el Spectrum.
- Conector para otros periféricos.

El Discovery se conecta simplemente en la parte trasera de su Spectrum y le proporciona acceso instantáneo a la velocidad, prestaciones y seguridad de un sistema de disco.

El Discovery está especialmente diseñado para aceptar los mismos comandos que el Interface 1, permitiéndole usar la mayoría de los programas de casette diseñados para Microdrive sin necesidad de modificarlos.

El Discovery no ocupa sitio en la RAM del Spectrum por lo que podrá seguir disfrutando de sus programas de juego favoritos sin necesidad de desconectarlo ya que deja libre toda la memoria del Spectrum. Además, puede ya guardar la fuente de alimentación del Spectrum puesto que el Discovery le proporciona tensión a través del conector trasero.

El Discovery, con acceso aleatorio permite, por fin, aplicaciones serias para el Spectrum; contabilidades, facturaciones, tratamientos de textos, etc., que eran posibilidades solamente, están ahora a su alcance con el Discovery. Hay a su disposición un amplio catálogo de programas realmente útiles y otros que van apareciendo continuamente, que le permitirán aprovechar la insospechada capacidad de su Spectrum.



SISTEMAS INCLUIDOS

Salida para impresora	Compatible Centronics
Interface joystick	Conector standard tipo D de nueve contactos. Compatible Kempston.
Salida para monitor	Conector standard para monitor monocromo
Conector de expansión	De 56 vías
Alimentación	220 V. 50 Hz. con cable e interruptor incorporado. No se precisa la alimentación del Spectrum.

FUTURAS AMPLIACIONES

El Discovery 1 puede ampliarse a Discovery 2 en cualquier momento, colocando otra unidad de disco idéntica en el espacio previsto para ello. Todas las conexiones ya están efectuadas en el interior para simplificar el montaje.

El Discovery Plus, que es el nombre del kit de ampliación, está formado por la unidad de disco, 1 cable de 34 vías y una RAM 6116, que va insertada en un zócalo previsto a este fin.

Discovery 1 - El nuevo poder que respalda a su Spectrum

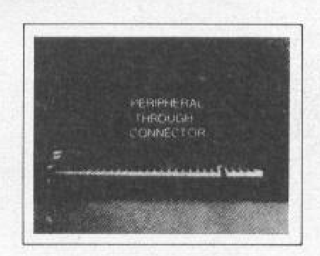


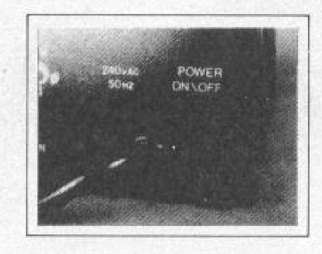
PROGRAMAS DISPONIBLES

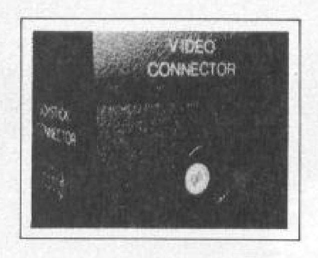
- Contabilidad profesional
- Procesador de textos
- Base de datos
- etc.....

UNIDAD DE DISCO

Tipo	Standard de 3.5"
Modelo	35401
Pistas	40
Caras	1
Capacidad total	250 Kbytes
Capacidad formateado	180 Kbytes
Tiempo de acceso pista/pista	3 milisegundos
Alimentación	La del Discovery









• Sistema RTTY (Rx-Tx) para Spectrum	4.800 Ptas.
• Programador de EPROMS (desde la 2716 a la 27128) *(para el Spectrum y el Amstrad 464)*	
• Interface Centronics para QL	12.800 Ptas.
• Impresora Ibico 80 columnas	30.400 Ptas.

Opus.
DISCOVERY 1

SOLICITELO A SU DISTRIBUIDOR DE INFORMATICA O DIRECTAMENTE A



SISTEMAS LÓGICOS GIRONA, S.A.
Avda. San Narciso, 24
17005 GERONA - Tel. (972) 23 71 00

por que en MODE 1 parpadearía también el caracter contiguo al cursor. EDCOPY copia la línea que se está editando en la parte baja de la pantalla cada vez que es necesario, después de cada nuevo caracter por ejemplo. Para ello utiliza OUTLI2 y después pone los espacios necesarios para asegurarse de que no queda ningún resto de la línea anterior, como podría ocurrir si, por ejemplo, acabamos de borrar un caracter. Siempre escribe un espacio de más para salvar el problema de los medios caracteres.

La rutina KEYIN es la encargada de interpretar los datos suministrados por KEYBOARD en LASTK y TVDATA.

La rutina KEYIN es la encargada de interpretar los datos suministrados por KEYBOARD en LASTK y TVDATA. LASK contiene el código que debe suministrar KEYIN en todos los casos excepto para los correspondientes a FLASH, BRIGHT e INVERSE. En estos casos la rutina debe calcular el caracter de control correspondiente, y dejar el parámetro en TVDATA para que sea entregado

```
        LD    (IY+2),^10
        CALL  CLALL
        JP    ^17A0
OUTLI2  SET   0,(IY+1)
        PUSH  DE
        EX    DE,HL
        RES   2,(IY+48)
        LD    HL,^5C3B
        RES   2,(HL)
        BIT   5,(IY+55)
        JR    Z,OUTLI4
        SET   2,(HL)
OUTLI4  LD    HL,(^5C5F)
        AND   A
        SBC   HL,DE
        JR    NZ,OUTLI5
        LD    A,^3F
        CALL  OUTINV
OUTLI5  CALL  OUTCUR
        EX    DE,HL
        LD    A,(HL)
        CALL  ^18B6

        INC   HL
        CP    ^0D
        JR    Z,OUTLI6
        EX    DE,HL
        CALL  OUTCH
        JR    OUTLI4
OUTLI6  POP   DE
        RET
OUTCH   CALL  ^2C88

        CCF
OUTCH3  JP    NC,^196C
        CP    ^20
        JR    Z,OUTCH3
        CP    ^2C
        JR    Z,OUTCH3
        CP    ^1A
        JP    ^193E
EDCOPY  XOR   A
        LD    (PRDAT+2),A
        CALL  ^0D4D

        RES   3,(IY+2)
        RES   5,(IY+2)
        LD    HL,(^5C8A)
        PUSH  HL
        LD    HL,(^5C3D)
        PUSH  HL
        LD    HL,EDFULL
        PUSH  HL
        LD    (^5C3D),SP

        LD    HL,(^5C82)
        PUSH  HL
        SCF
        CALL  ^1195
        EX    DE,HL
        CALL  OUTLI2
        EX    DE,HL
        CALL  OUTCUR

        LD    HL,(^5C8A)
        EX    (SP),HL
        EX    DE,HL
        CALL  ^0D4D
EDBLAN  LD    A,(^5C8B)

        SUB   D
        JR    C,EDDONE
        JR    NZ,EDSPAC

        LD    A,E
        SUB   (IY+B0)
        JR    NC,EDDONE

EDSPAC  LD    A,^20
        PUSH  DE
        CALL  PROUT
        POP   DE
        JR    EDBLAN

EDDONE  LD    A,^20
        RST   ^10
        JP    ^117C
EDFULL  LD    D,^00
        LD    E,(IY-2)
        LD    HL,^1A90
        CALL  ^03B5
        LD    (IY+0),^FF
        LD    DE,(^5C8A)
        INC   E
        JP    ^117E
OUTCUR  LD    HL,(^5C5B)
        AND   A
        SBC   HL,DE
        RET   NZ
        LD    A,(^5C41)
        RLC   A
        JR    Z,OUTC1
        ADD   A,^43

        JR    OUTC2
OUTC1   LD    HL,^5C3B
        RES   3,(HL)
```

YA ESTÁ A LA VENTA



PIDALA EN SU QUIOSCO HABITUAL O SOLICITELA A:

INFODIS, S.A.
Bravo Murillo, 377 • 28020 Madrid

VIRGIN-1-SPECTRUM

CUPON DE PEDIDO

Ruego me envien la cinta cassette **VIRGIN SOFTWARE** al precio de **525** pts.
El importe lo abonaré. Con mi tarjeta de crédito ☐ American Express ☐
Visa ☐ Interbank ☐
Contra reembolso ☐ Adjunto cheque ☐
Número de mi tarjeta ______
Fecha de caducidad ______
NOMBRE ______
DIRECCION ______
CIUDAD ______
C.P. ______
PROVINCIA ______
Sin gastos de envio

en la siguiente llamada a KEYIN. En el caso de cambio de color de papel o tinta el dato ya está en TVDATA, con lo que lo único que hay que hacer es asegurarse de que lo recogerá en la siguiente llamada. Esto se consigue cambiando la dirección de entrada del canal «K».

El ELSE marca el comienzo de una nueva sentencia.

Editor

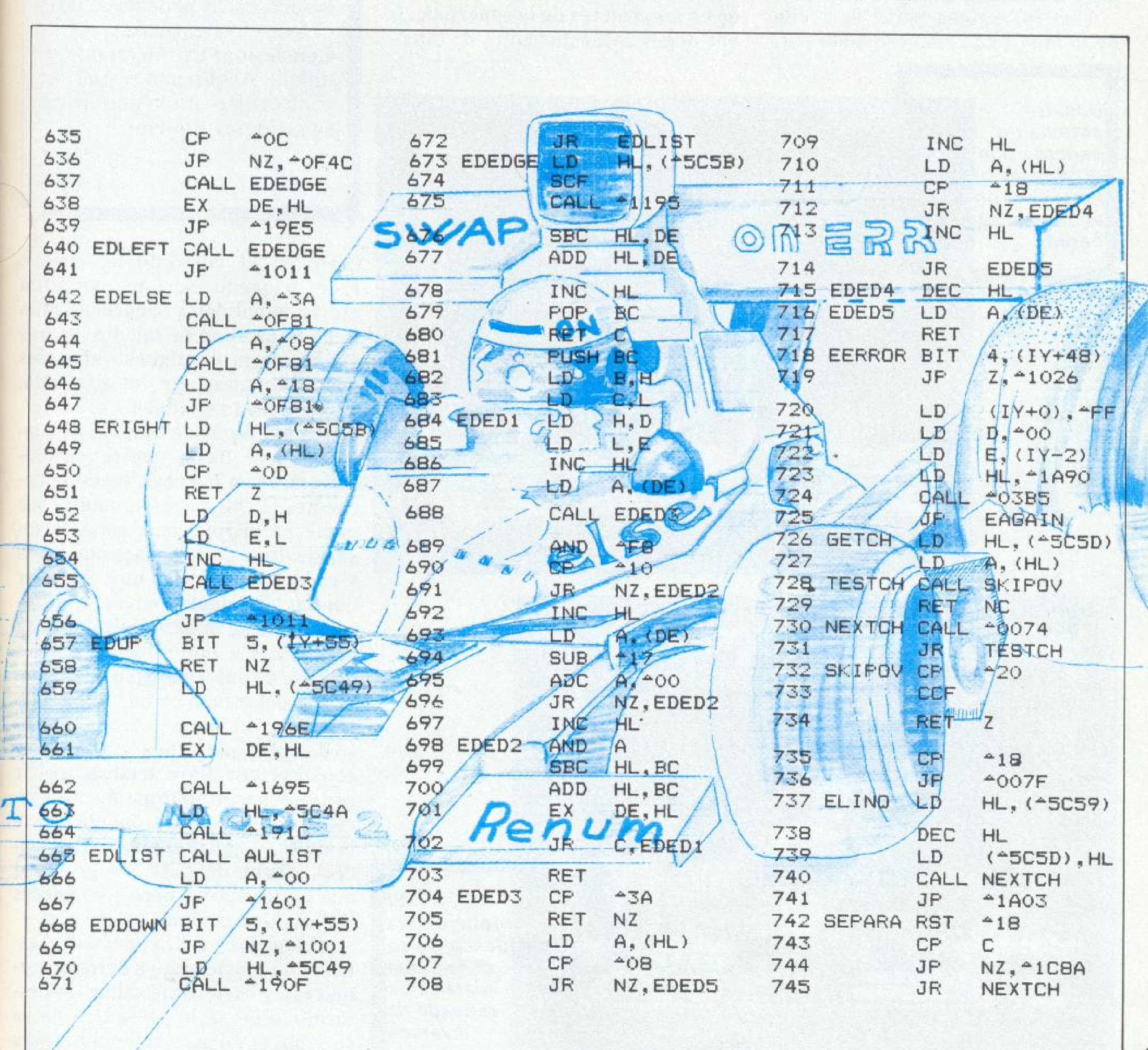
El EDITOR es el encargado de gestionar todo el proceso de introducción de una línea, o los datos de un INPUT. Para ello va leyendo los caracteres del teclado a través de WAIT-KEY y los interpreta. Si se trata de un caracter normal simplemente lo introduce en la línea, la lista y espera otro caracter. Un caso especial es ELSE. Este comando marca el comienzo de una nueva sentencia, y si apareciera sin más dentro de una línea sería problemático detectar el final de la sentencia anterior. Por ello cuando se pulsa ELSE en realidad se introducen tres caracteres en la línea, el

```
              DEC   DE
              LD    A,(DE)
              INC   DE
              CP    ^2C
              JR    Z,OUTC3
              CP    ^19
      OUTC3   LD    A,^4B
              JR    Z,OUTC4
              BIT   2,(HL)
              JR    Z,OUTC2
      OUTC4   SET   3,(HL)
              INC   A
              BIT   3,(IY+48)
              JR    Z,OUTC2
              LD    A,^43
      OUTC2   PUSH  DE
              CALL  OUTINV
              POP   DE
              RET
      OUTINV  EXX
              LD    HL,^5C91
              LD    D,(HL)
              PUSH  DE
              LD    (HL),^04
              CALL  PROUT
              POP   HL
              LD    (IY+87),H
              EXX
              RET
      CLS     CALL  CLALL
      CLSLO   CALL  ^0D6E
              JP    CLCHAN
      KEYIN   BIT   3,(IY+2)
              CALL  NZ,EDCOPY
              AND   A
              BIT   5,(IY+1)
              LD    A,(^5C08)
              RET   Z
              RES   5,(IY+1)
              PUSH  AF
              BIT   5,(IY+2)
              CALL  NZ,CLSLO
              POP   AF
              CP    ^18
              JR    NC,KDONE
              CP    ^10
              JR    NC,KCONTR
              CP    ^06
              JR    NC,KEYMCL
              LD    B,A
              AND   ^01
              LD    C,A
              LD    A,B
              RRA
              ADD   A,^12
              JR    KDATA
      KEYMCL  JR    NZ,KMODE
              LD    HL,^5C6A
              LD    A,^08
              XOR   (HL)
              LD    (HL),A
              JR    KFLAG
      KMODE   CP    ^0E
              RET   C
              SUB   ^0D
              LD    HL,^5C41
              CP    (HL)
              LD    (HL),A
              JR    NZ,KFLAG
              LD    (HL),^00
      KFLAG   SET   3,(IY+2)
              CP    A
              RET
      KDATA   LD    (IY-45),C
      KCONTR  LD    DE,KNEXT
              JR    KCHAN
      KNEXT   LD    A,(^5C0D)
              LD    DE,KEYIN
      KCHAN   LD    HL,(^5C4F)
              INC   HL
              INC   HL
              LD    (HL),E
              INC   HL
              LD    (HL),D
      KDONE   SCF
              RET
      EDITOR  LD    HL,(^5C3D)
              PUSH  HL
      EAGAIN  LD    HL,EERROR
              PUSH  HL
              LD    (^5C3D),SP
      EDLOOP  CALL  ^15D4
              PUSH  AF
              LD    D,^00
              LD    E,(IY-1)
              LD    HL,^00C8
              CALL  ^03B5
              POP   AF
              LD    HL,EDLOOP
              PUSH  HL
              CP    ^0A
              JR    Z,EDDOWN
              CP    ^0B
              JR    Z,EDUP
              CP    ^08
              JR    Z,EDLEFT
              CP    ^18
              JR    Z,EDELSE
              CP    ^09
              JR    Z,ERIGHT
```

primero en dos puntos, que marca el final de la sentencia, después viene un CHR$ 8 (BACKSPACE) y por último el ELSE. De esta forma la palabra ELSE se escribe encima de los dos puntos y éstos no aparecen. Las rutinas correspondientes a EDUP y EDDOWN han sido reescritas para evitar que llamen a AUTO-LIST en la ROM. EDLEFT, ERIGHT y DELETE están para poder saltar de un solo golpe los tres caracteres correspondientes a ELSE o los dos de un cambio de color, evitando el problema de que el cursor se meta entre ellos.

Cuando se pulsa ELSE en realidad se introducen tres caracteres en la línea.

La función de GETCH y NEXTCH es ir cogiendo los caracteres del programa utilizando CH-ADD como puntero. Ha sido reproducida por que la de la ROM no considera los códigos de los nuevos comandos y se los salta.




```
         CP    ^0C
         JP    NZ,^0F4C
         CALL  EDEDGE
         EX    DE,HL
         JP    ^19E5
EDLEFT   CALL  EDEDGE
         JP    ^1011
EDELSE   LD    A,^3A
         CALL  ^0F81
         LD    A,^08
         CALL  ^0F81
         LD    A,^18
         JP    ^0F81
ERIGHT   LD    HL,(^5C5B)
         LD    A,(HL)
         CP    ^0D
         RET   Z
         LD    D,H
         LD    E,L
         INC   HL
         CALL  EDED3
         JP    ^1011
EDUP     BIT   5,(IY+55)
         RET   NZ
         LD    HL,(^5C49)
         CALL  ^196E
         EX    DE,HL
         CALL  ^1695
         LD    HL,^5C4A
         CALL  ^191C
EDLIST   CALL  AULIST
         LD    A,^00
         JP    ^1601
EDDOWN   BIT   5,(IY+55)
         JP    NZ,^1001
         LD    HL,^5C49
         CALL  ^190F
         JR    EDLIST
EDEDGE   LD    HL,(^5C5B)
         SCF
         CALL  ^1195
         SBC   HL,DE
         ADD   HL,DE
         INC   HL
         POP   BC
         RET   C
         PUSH  BC
         LD    B,H
         LD    C,L
EDED1    LD    H,D
         LD    L,E
         INC   HL
         LD    A,(DE)
         CALL  EDED3
         AND   ^FB
         CP    ^10
         JR    NZ,EDED2
         INC   HL
         LD    A,(DE)
         SUB   ^17
         ADC   A,^00
         JR    NZ,EDED2
         INC   HL
EDED2    AND   A
         SBC   HL,BC
         ADD   HL,BC
         EX    DE,HL
         JR    C,EDED1
         RET
EDED3    CP    ^3A
         RET   NZ
         LD    A,(HL)
         CP    ^08
         JR    NZ,EDED5
         INC   HL
         LD    A,(HL)
         CP    ^18
         JR    NZ,EDED4
         INC   HL
         JR    EDED5
EDED4    DEC   HL
EDED5    LD    A,(DE)
         RET
EERROR   BIT   4,(IY+48)
         JP    Z,^1026
         LD    (IY+0),^FF
         LD    D,^00
         LD    E,(IY-2)
         LD    HL,^1A90
         CALL  ^03B5
         JP    EAGAIN
GETCH    LD    HL,(^5C5D)
         LD    A,(HL)
TESTCH   CALL  SKIPOV
         RET   NC
NEXTCH   CALL  ^0074
         JR    TESTCH
SKIPOV   CP    ^20
         CCF
         RET   Z
         CP    ^18
         JP    ^007F
ELINO    LD    HL,(^5C59)
         DEC   HL
         LD    (^5C5D),HL
         CALL  NEXTCH
         JP    ^1A03
SEPARA   RST   ^18
         CP    C
         JP    NZ,^1C8A
         JR    NEXTCH
```

NODES OF YESOD

SERMA
Spectrum 48 K
2.995 ptas.

Una misteriosa señal se recibe de la luna y tú eres designado para explorar las cuevas subterráneas de donde procede, con la misión de encontrar su origen. Para ayudarse en la búsqueda necesitas una pequeña criatura lunar que se esconde en los cráteres de la superficie, y que debes conseguir antes de internarte en el laberinto de cuevas. Este pequeño ser te permitirá mayor movilidad y seguridad, pues tiene la facultad de taladrar muros y defenderte de algunas alimañas que intentarán ir desgastando, poco a poco tu vitalidad.

Control: Teclado y Joystick.
Jugadores: Uno.
Gráficos: Excelentes, tanto en la animación como en el entorno.
Nivel de dificultad: Bastante alto, se mantiene durante todo el juego.
Originalidad: El juego está basado en la película «2001 Odisea en el espacio».
Conclusión: Es interesante y difícil. Aconsejamos que se confeccione un plano para su mejor seguimiento.



Nuestro astronauta espera a que la criatura lunar cave un agujero en la pared.



El protagonista descansa en una plataforma rodeado de diversos peligros.

Tus instrucciones son las de conseguir ocho llaves que te permitirán entrar en la cueva donde se encuentra el enigmático monolito que nadie ha visto jamás; estas llaves son de diversas formas geométricas y cada vez que cojas una, un color que las distingue aparecerá en los cuadrados de la parte inferior de la pantalla. Un indicador de señales de vida te informará en todo momento del estado en que te encuentras, a la vez que te indica el número de vidas que tienes. Una vez que consigas una llave tendrás que ir con precaución porque puede aparecer un extraño astronauta que te la quita, aunque algunas veces podrás clavar un poste de gravedad que lo neutraliza, pero sólo puedes utilizarlo en contadas ocasiones.

El jugador puede controlar el astronauta moviéndolo a derecha e izquierda y haciéndole saltar los obstáculos que se le presenten en su camino; tanto para bajar como para subir por un precipicio tendrá que saltar de una plataforma a otra.

También controla los movimientos de la criatura lunar desplazándola en el espacio en todas las direcciones.

Los gráficos son excelentes, destacando, sobre todo, las figuras del astronauta y de la criatura lunar y sus movimientos. En cuanto al sonido, diferencia cada una de las acciones con un efecto distinto, siendo un aviso para el jugador en algunas ocasiones de peligro. Si acaso te perdieras o deseas descansar puedes hacer una pausa pulsando la tecla (ENTER). Al final del juego aparece en la pantalla el tanto por ciento realizado y el tiempo empleado.

HIGHWAY ENCOUNTER

ABC Soft
Spectrum 48 K
1.900 ptas.



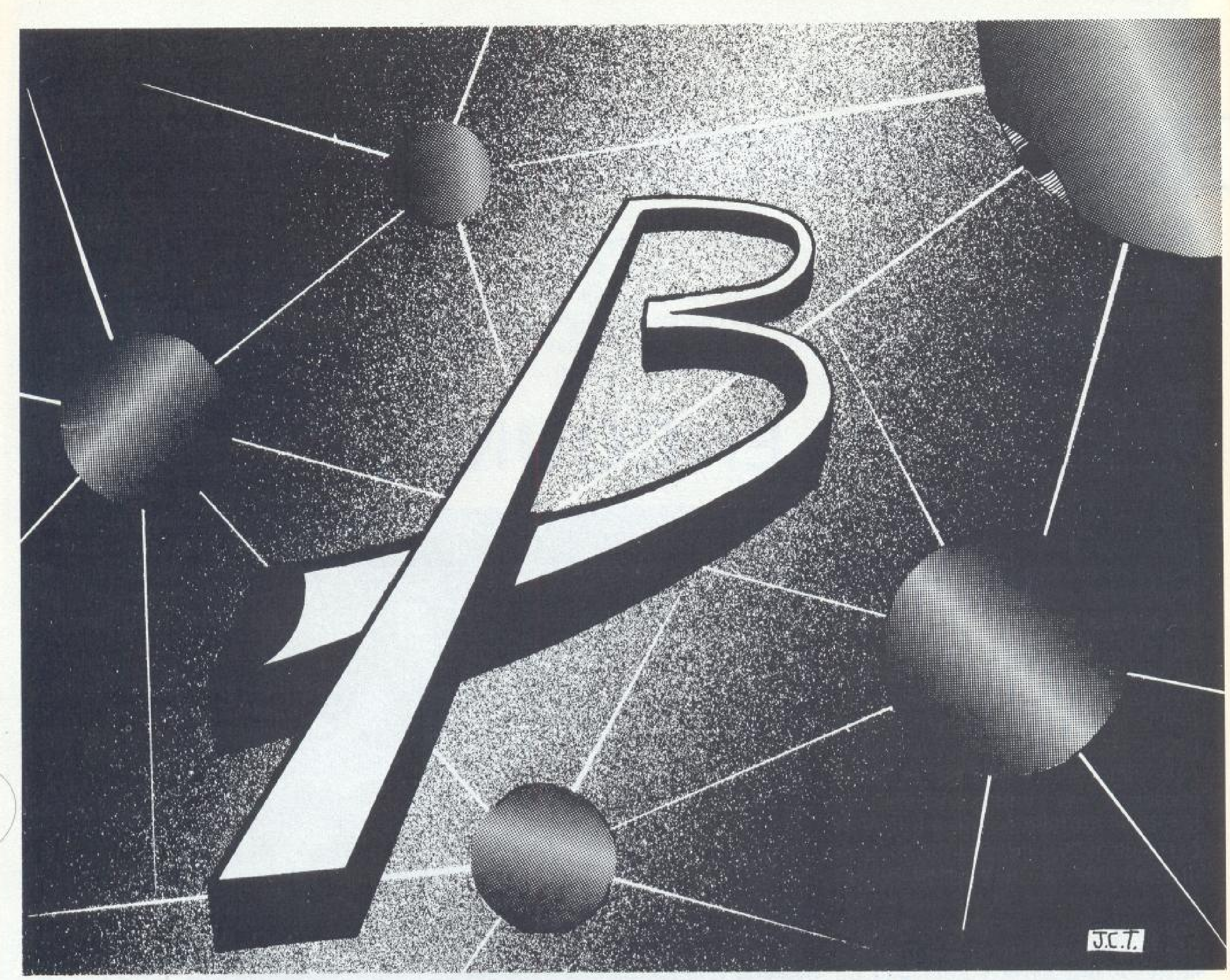
Nuestro láser robot activo se enfrenta a dos obstáculos y dos llamas.

Highway Encounter es un desafío a todos aquellos que se consideran muy hábiles en el dominio de los video juegos. Característica importante de este juego son sus gráficos realizados en tres dimensiones, que junto con los colores empleados hacen de él un programa atractivo que incita a seguir jugando.

Con la primera dificultad que te encuentras es que los cinco láser robots (las vidas) de que dispones salen en fila india y sólo uno de ellos tiene movilidad, esto supone, en principio, un inconveniente puesto que dificulta los movimientos del robot en activo que debe abrir camino a los demás que sólo se mueven en dirección recta y sin posibilidad de detenerlos hasta que tropiezan en algún obstáculo. Si el robot que controlas es destruido, inmediatamente uno de la fila lo sustituye.

La fila de los cuatro láser inactivos por un obstáculo.

Control: Teclado y Joystick.
Jugadores: Uno.
Gráficos: Buenos gráficos en tres dimensiones.
Nivel de dificultad: Aumenta a medida que se avanza.
Originalidad: Bastante original, sobre todo porque juegas con todas las vidas desde un principio.
Conclusión: Juego difícil que requiere mucha agilidad y rapidez al controlar el robot.



BETA BASIC: 50 nuevos comandos

Muchos de los detractores ael Spectrum alegan que tiene un dialecto BASIC de lo más peculiar. No vamos a entrar en discusión sobre las ventajas e inconvenientes de la versión BASIC que Sinclair introdujo en sus ordenadores, pero para acallar totalmente a estos críticos existe un programa que resulta un instrumento útil en todos los ámbitos, dando al Spectrum un alcance comparable con el de ordenadores mucho más potentes. Este programa se llama Beta BASIC.

En España hay al menos dos versiones, lógicamente mucho más potente la segunda, aunque ya la primera supone un gran avance respecto al ordenador estándar. No obstante, nos vamos a referir generalmente a la versión reciente, dado que es más completa.

El programa tiene dos partes: una corta, con 3 líneas BASIC. Se definen varias funciones nuevas en la línea 0, que nos acompañará en todo nuestro trabajo; la línea 1 actúa como copiador de seguridad y una tercera línea, la 2, que inicializa el programa. La segunda parte son *bytes,* aproximadamente 9K que se colocan en la parte alta de la memoria reservada mediante el CLEAR de la línea 2. Una vez cargado el programa empezaremos a notar diferencias: la primera el mensaje de presentación, que aparecerá como:

Beta BASIC 1.8 Betasoft 1984

En seguida vamos a notar una diferencia con el funcionamiento normal del Spectrum, y es el sonido del teclado, que se hace notar mucho, con un tono distinto. No obstante, es posible eliminarlo con un POKE en la dirección 23609, poniéndolo a cero. También persistirá la línea cero a pesar de desaparecer las líneas 1 y 2.

Una gran ventaja que posee es su compatibilidad con programas escritos sin éste, con la única precaución de usar MERGE en lugar de LOAD para no perder la línea 0 y pulsar KEYWORDS 0 si va a haber gráficos definidos por el usuario.

Con este programa se introducen una serie de comandos que normalmente sólo están disponibles en ordenadores profesionales. También se introducen comandos BASIC típicos que el Spectrum no posee, funciones útiles, etcétera. Lo cierto es que el catálogo es bastante extenso. Iremos desglosando los más importantes. En otros, dado el poco espacio disponible, daremos una escueta indicación de su uso.

El manual que acompaña a este programa es lo suficientemente claro y extenso, a pesar de parecer una traducción literal del inglés, como para comprender el funcionamiento del programa, así como todas las novedades que introduce.

Con las nuevas instrucciones aparece un problema: los malabarismos que hay que hacer para seleccionarlos. Los comandos (DO, EXIT IF, DEF PROC, etcétera), están almacenados como gráficos definidos. Para llamarles, se hace igual que para un gráfico, pulsamos CAPS SHIFT + GRAPHICS si tenemos un Spectrum normal o directamente GRAPHICS en un Plus, apareciendo un cursor indicativo y pudiendo pulsar el comando que deseemos, del que debemos conocer su situación en el teclado, generalmente alfabética. Por ejemplo, CLOCK se encuentra en la letra C. Este es uno de los pocos puntos criticables del programa: hasta después de cierto tiempo de uso, no acabamos de encontrar los comandos. Si queremos gráficos definidos por el usuario, debemos pulsar KEYWORDS 0, y hasta no restablecer KEYWORDS 1 no veremos el formato normal de los nuevos comandos, aunque sigan funcionando.

En cuanto a las funciones, su escritura es más fácil, ya que es igual que en el Spectrum normal, con la particularidad de que escribimos por ejemplo:

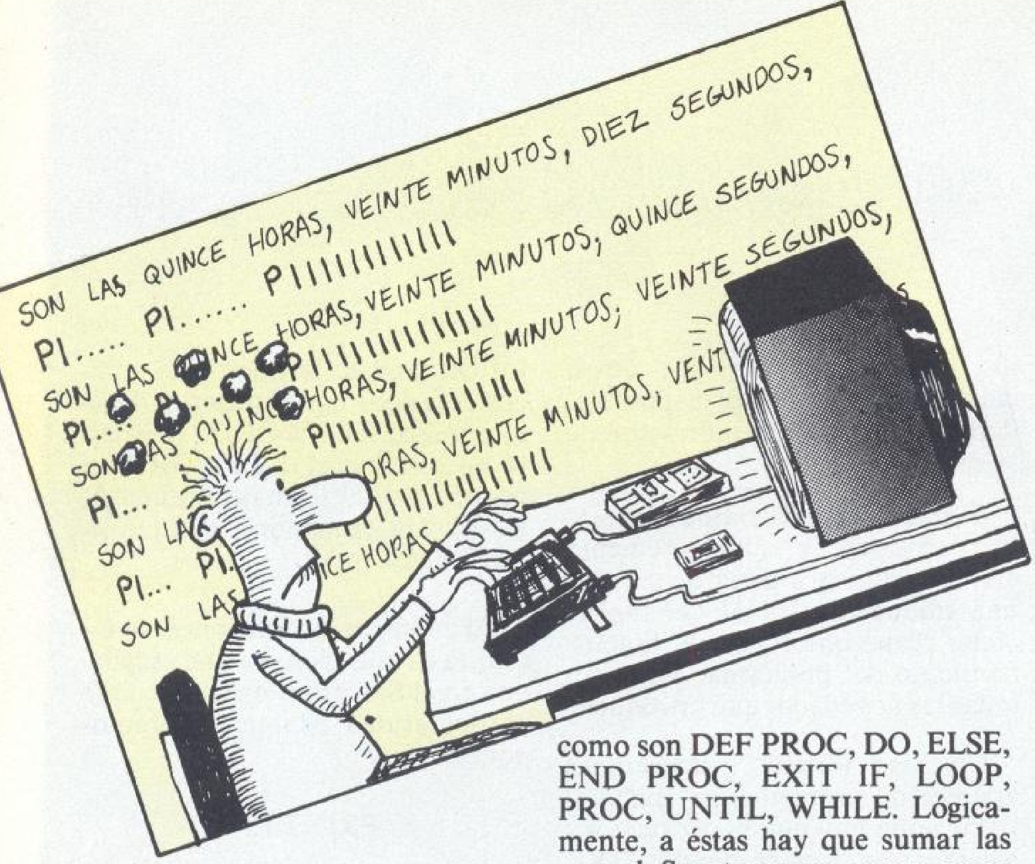
PRINT FN

y al añadir la letra de la función seguida de $ si lo necesita, desaparece FN y la letra, apareciendo el nombre de la función deseada, por ejemplo, pulsando U$, aparecerá al final:

PRINT USING$

y a partir de aquí, seguiremos normalmente con la sintaxis propia de la función. Hay que notar que en total, se introducen 21 nuevas funciones, algunas de ellas como COSE, SINE, RND, mejoras de las ya disponibles. No obstante, el usuario puede definir las suyas propias, siempre y cuando no se solapen con las del Beta BASIC, que no son borrables.

Un comando que supone una mejora de la segunda versión respecto a la primera es CLOCK, que como indica su nombre es un reloj. Podemos indicarle una hora nada más empezar, o él echará a andar desde las 00:00:00, al arrancar el programa. Es posible usarlo como alarma. Esta puede avisarnos a la hora deseada o bien ir a una subrutina prefijada. Es ideal para programarnos el tiempo que queremos pasar delante del ordenador. Tiene 7 opciones dependiendo de si queremos tener alarma, que aparezca en *display* continuamente o que vaya a una subrutina al llegar la hora.



Definición de teclas: algo que hacía falta

Otra opción que equipara al Spectrum con otros ordenadores es la posibilidad de definición de las teclas. Por ejemplo, supongamos que a menudo pulsamos LIST 3 TO 10. Se puede hacer que al pulsar sólo una tecla, por ejemplo, la L, se ejecute esto. Para ello, es necesario teclear:

DEF KEY "L": "LIST 3 TO 10"

Muchos dirán en este momento: «En mi Spectrum no existe una instrucción que se llame DEF KEY.» Es cierto, pero sí poseen este programa y ponen el cursor en modo gráfico, pulsan SYMBOL SHIFT y la tecla 1, verán qué pronto les aparece.

Podemos definir 21 nuevas funciones de este tipo, que con el tiempo veremos que son un arma potente.

Beta BASIC mejora la estructura de nuestros programas

En este apartado el Beta BASIC nos ayuda con varias instrucciones como son DEF PROC, DO, ELSE, END PROC, EXIT IF, LOOP, PROC, UNTIL, WHILE. Lógicamente, a éstas hay que sumar las que el Spectrum ya posee como son IF, GOSUB, etcétera.

Podemos empezar con DEF PROC y sus relacionadas, END PROC y PROC. Son abreviaturas de DEFine PROCedure, o sea, definición de procedimiento. Es una herramienta muy útil, ya que se le puede comparar con una subrutina con la ventaja de que puede ser localizable en cualquier parte sin tener que invocar el número de línea donde está al llamarla. Se le debe dar un nombre, con lo que en principio, el número de PROCedures posibles está restringido sólo por la memoria que tengamos. Son muy útiles y ordenadores mucho más potentes como el QL confían en esta estrategia.

Pongamos un ejemplo: supongamos que varias veces inicializamos una parte de un programa borrando la pantalla, pidiendo el nombre y almacenándolo. Podemos hacer como se muestra en el listado 1.

Las posibilidades de esta instrucción son infinitas, este es sólo un ejemplo, en el que nos hemos librado de escribir GOSUB (núm. de línea), sin más ventaja. En programas con mayor estructuración, es posible aumentar la legibilidad de nuestro programa.

Pasemos ahora a hablar de las instrucciones DO, EXIT IF, UNTIL, WHILE, LOOP. Viendo la primera y la última podemos adivinar que se trata de instrucciones de bucles *(loops)* con condiciones de salida. Pongamos un ejemplo. Supongamos que somos el empleado de un casino en el que hay un señor, cuyo DNI tenemos registrado, al que no debemos dejar pasar. Como a toda persona que va a estos locales se le pide el DNI para registrarlo, podremos utilizar un programa con esta condición. Con el listado 2 ó 3 podemos resolver

LISTADO 1

```
............ (programa)
1000 FOR N=1 TO NUMJUG
1010 PROC def              (mientras que PROC se obtiene con
                           la tecla 2, def se deletrea)
1020 NEXT N
............ (RESTO DE PROGRAMA)
............
XXXX DEF PROC def
        CLS: INPUT "Tu nombre?": a$ (N)
        END PROC
```

este problema (por simplicidad, y dado que en programación estructurada el número de línea es simbólico, los omitimos).

En este listado hemos visto que con EXIT IF salimos del bucle siempre que se desee. La instrucción UNTIL es contraria a WHILE, ya que hará lo prescrito hasta (UNTIL en inglés) que se cumpla una afirmación lógica. Un listado de aplicación de este comando, sería al principio:

```
DO
INPUT "DNI"
DO UNTIL DNI = (NUM.)
```

resto del listado igual que el primero.

El BASIC del Spectrum no tiene ya nada que envidiar

Pasemos ahora al segundo bloque, después de introducir ligeramente estos comandos de programación estructurada, donde vamos a ver instrucciones típicas de Basic que otros ordenadores poseen y se echaban de menos en el Spectrum. Dentro de estos comandos se pueden destacar ELSE, GET, ON, ON ERROR, etcétera.

El comando ELSE es muy típico de otros ordenadores y ahora lo tenemos en el Spectrum. La única precaución a tener en cuenta es poner antes el símbolo ":".

En cuanto a GET, es como una mezcla de INKEY$ e INPUT ya que espera una tecla, y no sigue hasta que no se pulse.

El comando ON, entre otras funciones, sirve para hacer un salto de línea dependiendo del valor de una variable. En el Spectrum se puede hacer con una expresión alfanumérica del tipo:

GOTO 1000*S+12

mientras que en la mayoría de los ordenadores domésticos esta ins-



trucción causa un Syntax Error, debiendo hacerse de la forma:

ON K GOTO 1012, 2012, etc.

dependiendo del valor de K, irá a la primera dirección, a la segunda, etcétera. Ahora el Spectrum posee las dos variedades.

En cuanto al comando ON ERROR; va a ser muy útil cuando escribamos un programa donde estemos preocupados por la posible aparición de errores. Por ello, si ponemos ON ERROR podremos ir a una subrutina preestablecida que podrá corregir el error o indicarlo al menos. El formato de la instrucción es:

ON ERROR (núm. de línea)

yendo directamente a la línea especificada en cuanto se detecte un error.

Otro comando que resulta muy mejorado con este programa es LIST, del que estamos acostumbrados a poner un número de línea. Bien, ahora podemos poner además el número de línea hasta la que queremos listar. Es decir, es posible hacer:

LIST 10 TO 50

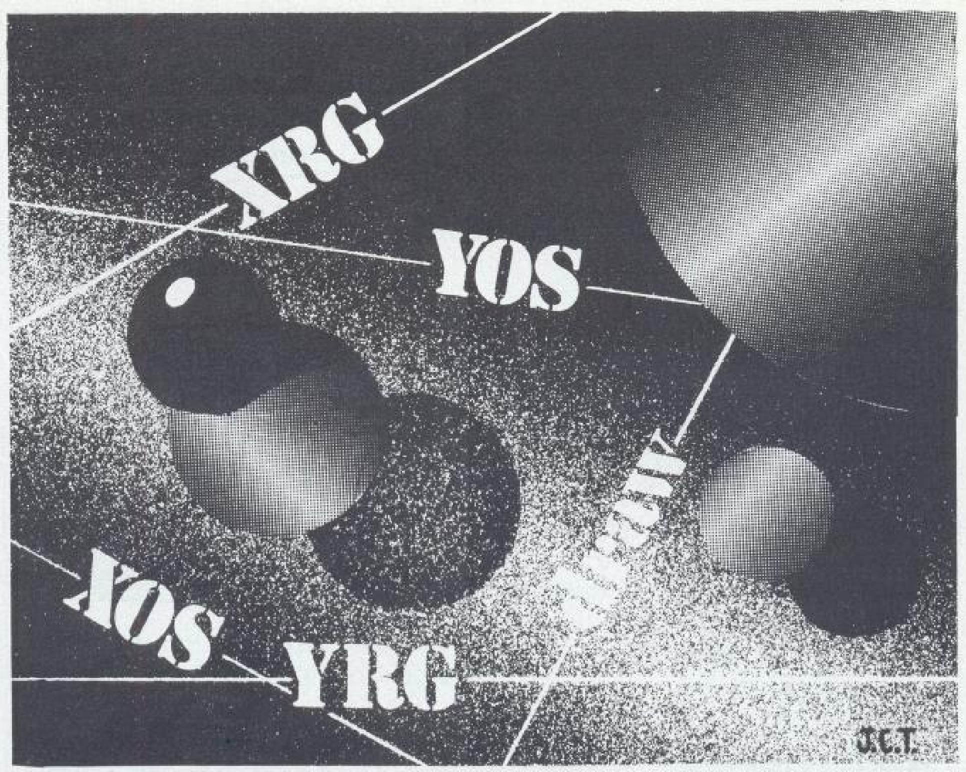
Otro comando a analizar es POP, que muchos usuarios de Spectrum se han hecho en código máquina, que saca del *stack* la dirección de vuelta de subrutina para ir a otra dirección diferente a donde se llamó.

También aquí podemos introducir el comando USING, que se echaba de menos. Tiene las mismas (o más) opciones que en ordenadores profesionales, siendo de lo más útil cuando tengamos que presentar datos con un formato normalizado. Se dispone de dos funciones de este tipo, USING y USING$, esta última en forma de

LISTADO 2

```
    DO
    INPUT DNI
    DO WHILE DNI < > (NUM FATIDICO)
    NUMOR=NUMOR+1
    A (NUMOR)=DNI
    INPUT "Su nombre, por favor?":n$
    LPRINT NUMOR, N$                    (escribimos su carnet)
    PRINT "FELIZ ESTANCIA SR.":N$;
    INPUT DNI
    LOOP
    PRINT "ACCESO DENEGADO"
    LOOP
```

Con el primer listado hemos puesto de manifiesto tanto como que pueden tenerse bucles dentro de otros bucles como el funcionamiento de la instrucción WHILE (del inglés mientras).



función definida (con FN). La primera la utilizaremos con expresiones numéricas y la otra con las alfanuméricas. Veamos un ejemplo de aplicación. Supongamos que estamos haciendo la contabilidad de una empresa donde las cantidades mayores tienen cuatro cifras enteras y dos decimales. Los datos tabulados se podrán conseguir con:

PRINT USING "####.##"; NUM

si núm=1002.234, aparecerá en pantalla 1002.23.

Es posible además introducir delante caracteres o números como 0 ó $. Si nuestro formato le viene pequeño al número que queremos imprimir, aparecerá un informe de *overflow* (desbordamiento), apareciendo el signo %. No se puede, sin embargo, utilizar notación científica. La función USING$ funciona de la misma forma pero en vez de número, se pone una cadena alfanumérica.

En cuestión de gráficos, nuestro Spectrum mejorará notablemente, con la posibilidad que da el nuevo comando PLOT de posicionar textos. Además, es posible modificar las escalas con XOS, XRG, YOS e YRG. Estas palabras no son comandos, sino variables, que se pueden modificar, pero no se pueden borrar con CLEAR o RUN, sino que se ponen a cero. Se utilizan para mover el origen de coordenadas hasta cualquier punto de la pantalla, modificando YOS y XOS, mientras que cambiando XRG e YRG modificaremos PLOT y DRAW.

Dentro de las funciones definidas agrupables en este bloque, podemos distinguir la función AND, que hace un AND lógico *bit* a *bit*, siendo diferente al AND Basic. Esta es una función que muy pocos ordenadores poseen. Para entender este comando, vamos a poner dos ejemplos, indicando previamente que en circuitería digital, la función AND dará salida 1 cuando los dos *bits* enfrentados sean 1. Supongamos los números 7 y 3 en principio. El resultado tendrá *bits* a 1 cuando se enfrenten dos *bits* que estén a 1.

00000111 = 7 entrada 1
00000011 = 3 entrada 2
00000011 = 3 salida

En este mismo contexto encontramos la función XOR, que también es *bit* a *bit*, o la función OR, igualmente.

Muchos ordenadores son capaces de pasar números a diferentes bases, sobre todo a binario y hexadecimal. Esto ahora es posible con las funciones BIN$ y HEX$, dando números binarios o hexadecimales o DEC, que pasa de hexadecimales a decimales.

Comandos complementarios al editor

Podemos distinguir AUTO, DELETE, EDIT, RENUM, etcétera, y aunque son muy conocidas vamos a dar una idea somera de su funcionamiento. La función AUTO sirve para escribir programas dándonos el ordenador el número de línea, partiendo de la que queramos y con el incremento que deseamos.

Un problema que muchos usuarios tienen a la hora de depurar programas es el borrar un cierto número de líneas. Con este comando se soluciona indicando:

DELETE (LINEA) TO (LINEA)

borrándose las líneas comprendidas entre los números introduci-

LISTADO 3

```
DO
INPUT "DNI Y NOMBRE, POR FAVOR": DNI, N$
EXIT IF DNI=(NUM. NON GRATO)
NUMOR=NUMOR+1
A (NUMOR)=DNI
LPRINT NUMOR. N$
PRINT "FELIZ ESTANCIA"
LOOP
```

dos. El comando EDIT también mejora mucho y sólo va a ser necesario pulsar 0 para que aparezca EDIT, pondremos nuestro número de línea y aparecerá la línea en la parte de abajo de la pantalla como antes ocurría. Otra sorpresa agradable que vamos a tener es que si pulsamos el cursor de la flecha hacia abajo y la línea editada ocupa dos líneas al menos, bajaremos a la línea de abajo, mientras que antes debíamos avanzar secuencialmente con la flecha derecha. RENUM también va a ser muy útil cuando queramos lavar la cara al programa dando a los números de línea una distancia fija, todo ello fijado por nosotros.

Muchos ordenadores incorporan una función muy útil, ya que ayuda a seguirle la pista al programa. Nos estamos refiriendo al comando TRON y su complementario TROFF. Mientras que en la primera versión de Beta Basic sí aparecen, en la segunda aparece una mejoría, ya que en su lugar disponemos de la función TRACE, que incorpora algunas ventajas. Estas funciones van indicando en pantalla el número de línea que van ejecutando pero además TRACE escribe la sentencia correspondiente. No obstante, el manual da una serie de recomendaciones para fabricarse a medida una rutina TRACE que podemos almacenar en una tecla definida como hemos visto antes.

Comandos o funciones relacionados con la memoria

Dentro de este apartado podemos distinguir DPOKE y DPEEK. Cuando se tiene un número entero menor de 65355 que queremos meter en memoria, tenemos que



meter el *byte* menos significativo en primer lugar y el otro en segundo lugar. Pues bien, esto nos lo soluciona directamente DPOKE y DPEEK.

También dentro de este apartado podemos introducir otras dos funciones más o menos complementarias JOIN y SPLIT. Supongamos que tenemos el siguiente listado:

```
10 PRINT "HOLA"
20 INPUT "Cuál es tu nombre?"; in$
```

y queremos para compactar el programa unir estas dos líneas. Con el comando JOIN va a ser muy simple, sólo debemos escribir:

```
JOIN 10
```

y la línea siguiente a 10 se unirá a ésta perdiéndose la línea 20 como tal, apareciendo:

```
10 PRINT "HOLA": INPUT "Cuál es tu nombre?": n$
```

Por el contrario, si al escribir el programa pensamos que una sentencia dentro de una línea es posible que la necesitemos en otra línea posteriormente podremos utilizar el comando SPLIT que en realidad no es una palabra clave, ya que lo que se introduce antes de la sentencia que pensamos dividir es el símbolo "< >".

Vamos a poner un ejemplo aclaratorio. Suponiendo la línea:

```
20 GOSUB 234: REM PRUEBA: < > PRINT "FUNCIONA"
```

al pulsar ENTER para subir la línea, aparecerá arriba

```
20 GOSUB 234: REM PRUEBA
```

mientras que abajo habrá quedado

```
20 (cursor) PRINT "FUNCIONA"
```

que podremos llevar a cualquier zona del programa.

Otra de las funciones definidas que Beta Basic trae nos indica la memoria libre que nos queda. Esto lo hace la función MEM, que introduciremos como FN M aunque después cambie a MEM, como vimos que hacían las funciones definidas.

Hemos dejado para lo último de este apartado la función SORT, que lo que hace es ordenar datos, a unas velocidades realmente prodigiosas. Es capaz de ordenar cadenas, números o letras tanto en orden descendente como ascendente de toda o parte de las variables que deseemos.

Comandos relacionados con el cálculo

Citamos a continuación tres rutinas que suponen una ventaja respecto a las que el Spectrum posee. Son COSE, SINE y RNDM. COSE y SINE son versiones del coseno y

seno que ya disponemos con la ventaja de ser sustancialmente más rápidos. En cambio, RNDM es una función RND mejorada.

Comandos relacionados con la pantalla, gráficos y color

Vamos a hablar de ALTER, FILL, ROLL, SCROLL y FILLED.

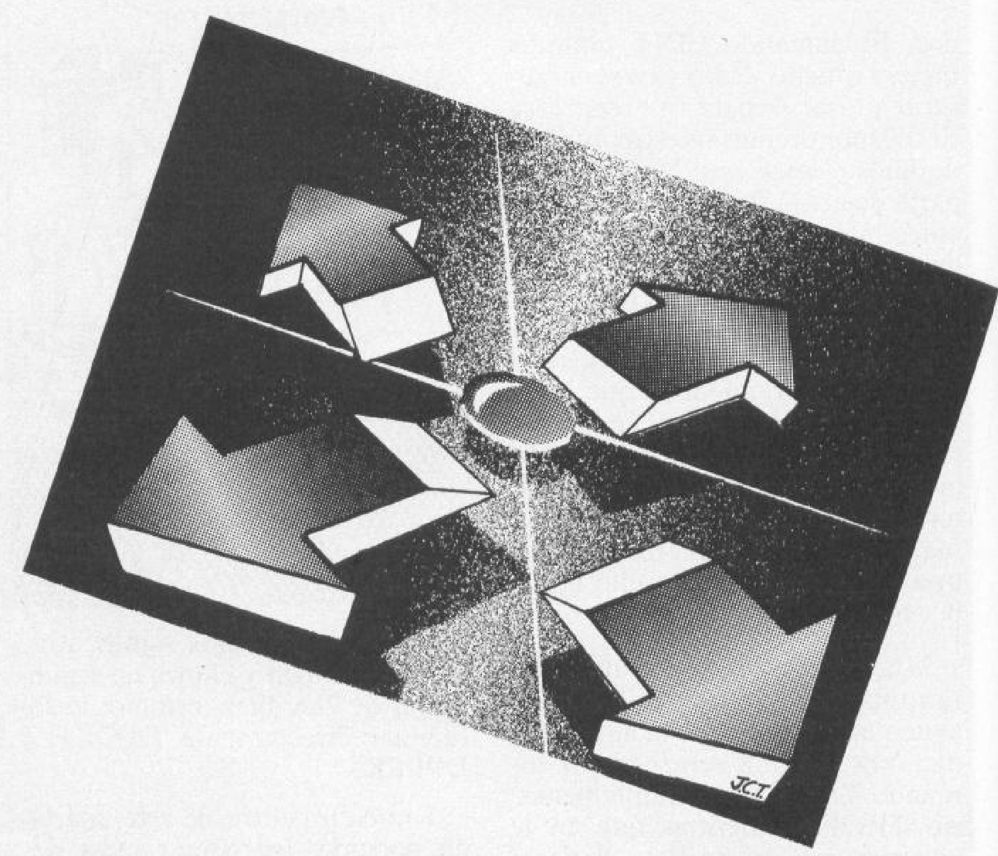
Empecemos por ALTER: Con él podremos cambiar los atributos de color de la pantalla.

ALTER TO PAPER 2

nos llevará a un color de papel rojo. Tiene muchas posibilidades y ventajas respecto al estándar del Spectrum, que sigue funcionando igualmente.

En cuanto a FILL lo que hace es rellenar zonas cambiando el color del papel por la tinta o al revés dependiendo de si se pone después de FILL PAPER o INK.

Vamos a pasar a ROLL y SCROLL, que son los comandos estrella de este apartado, ya que como es fácil de imaginar nos van a servir de *scroll* de ventanas previamente definidas por nosotros. Con ROLL es posible mover en cualquier dirección, definida por nosotros el contenido de la pantalla o de una ventana, con la ventaja de no perder la información que vaya saliendo de los márgenes que aparece por el lado contrario. ROLL efectúa un pequeño movimiento y para obtener efectos de continuidad se debe utilizar en un bucle para desplazar la ventana. En este comando podemos definir el número de *pixels* que queremos se desplacen. Pueden aparecer problemas con los atributos si no se considera bien el número de *pixels* a mover, aunque con cuidado se puede introducir en juegos escritos en Basic, obteniendo muy buenos resultados.



En cuanto a SCROLL desplaza la pantalla entera, pero mejorando lo que el Spectrum hace directamente, ya que es posibles desplazarla en las cuatro direcciones de los cursores, usándose los códigos de cursor para indicar la dirección. Así, si escribimos

SCROLL 6

la pantalla se moverá un *pixel* en la dirección que marca el cursor de la tecla 6, esto es, hacia abajo.

Una ampliación muy potente

De esta descripción podemos deducir que este programa potencia el lenguaje Basic del Spectrum, al que sin eliminar prácticamente nada añade gran número de comandos. Es quitando los cada vez más espectaculares juegos, uno de los programas más útiles que podemos tener para nuestro ordenador y que especialmente recomendamos como imprescindible para aquella persona que no tenga conocimientos suficientes como para hacerse rutinas de código máquina a su medida y utilice el Basic en sus aplicaciones.

No obstante, lo mejor para llegar a comprobar su potencia es la experimentación sin preocuparnos de que el sistema pueda colgarse, ya que, aunque no ha habido antes ocasión de comentarlo, la tecla BREAK ha sido modificada con un mecanismo de RESET que decide, después de un segundo de ser pulsada que el Spectrum debe andar colgado y hace un RESET.■

Alfonso Martín



Discos: más memoria a menos precio

nuevas utilidades: BCPL, Assembler, Monitor y Toolkit



NOTICIAS

ENSAMBLADOR

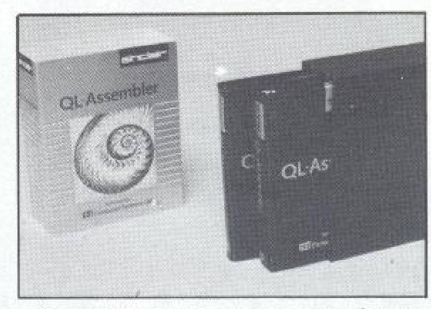
Es curioso que, en cualquier ordenador personal que se precie, uno de los primeros lenguajes que se implementan sea el ensamblador, cuando es ciertamente uno de los menos y peor usados. Sin embargo, la versión que vamos a comentar llega con un cierto retraso al mercado, ya que existen al menos dos versiones anteriores. A cambio, sale bajo el apoyo de Sinclair Research Ltd., lo que siempre es una garantía de un mínimo número de ventas.

El sistema ensamblador al que nos referimos ha sido desarrollado por **GST Computer Systems** para Sinclair, y se engloba dentro del primer conjunto de aplicaciones comercializadas bajo su nombre para el QL.

Se trata de un ensamblador sencillo, ya que no dispone ni de posibilidad de definición de macros ni de ensamblado condicional, pero, por otra parte, dispone de tres características de interés.

La primera de ellas es el conjunto de comandos para el listado del ensamblador, que permiten con facilidad documentar de un modo estético el desarrollo de un programa, lo que nunca es despreciable.

Otra característica tiene que ver con el uso de librerías de dos tipos. Por un lado, las clásicas librerías de rutinas en ensamblador que se leen desde el código fuente principal y se ensamblan en el momento y que permiten la creación de programas sumamente largos de los que normalmente no hubiera cabido todo el fuente ni en el editor ni en el ensamblador, y por otro lado, las librerías precompiladas, donde podemos disponer de rutinas muy útiles y frecuentemente utilizadas, que no ralentizan el tiempo de ensamblado, ya que ya han sido ensambladas previamente.



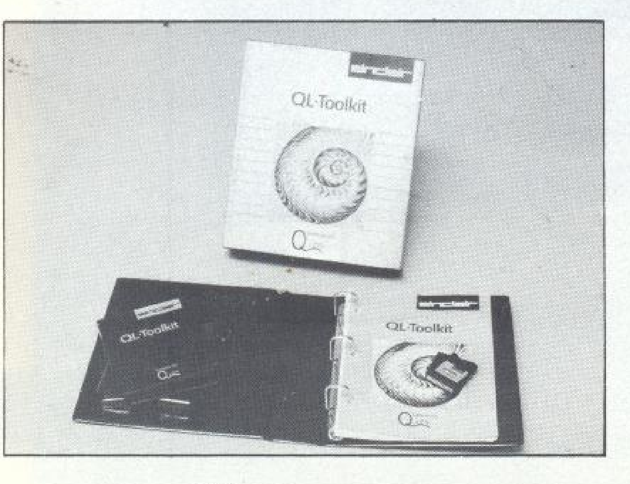
La última de estas ventajas es la existencia de mensajes de aviso, además de los de error, ya que sirven para indicar al programador dónde pudiera haber un error por haber otro modo mejor de realizar una operación, pero que no consisten en un error de concepto.

Hay, sin embargo, un gran inconveniente, y es la carencia de definiciones externas, que generan un tipo de archivos que han de ser reprocesados junto con otros del mismo tipo por medio de un programa especial llamado LINKER.

En resumen, se trata de un ensamblador sencillo y, por tanto, es para gente con ideas sencillas que no tenga ganas de complicarse la vida.

HERRAMIENTAS DE TRABAJO

Cuando empezamos a programar en el Superbasic de QL, echamos en falta gran cantidad de comandos y/o utilidades que nos harían la vida más fácil. Como la casa Sinclair, ciertamente, no es tonta, se ha preocupado de lanzar dentro de su paquete comercial inicial un conjunto de rutinas bajo el nombre de **Toolkit,** cuyo principal interés radica en que contiene precisamente todo lo que le falta al QL (o casi todo).



Las rutinas incluidas se pueden dividir en tres bloques. Un primer bloque es el de las extensiones al BASIC. Dentro de éstas cabe destacar el tan deseado editor de pantalla completa, que hasta ahora llevaban todos los ordenadores de precio medio menos el QL. Otros añadidos de interés son los que controlan el funcionamiento en multitarea, añaden el acceso aleatorio a los archivos, copian archivos sin interrumpir al BASIC, cambian de base numérica y otra serie de funciones interesantes.

El segundo bloque está formado por las rutinas de multitarea, entre las que se encuentran varios modelos distintos de relojes, digitales y analógicos, y una alarma horaria (para los que se despistan cuando tienen un teclado delante). Otras rutinas cambian el formato de los archivos, paginándolos o poniéndolos en mayúsculas, por ejemplo.

El último bloque es el formado por varios programas útiles escritos en Superbasic. Unos se encargan de la copia repetida de uno o más archivos a uno o más destinos distintos, otro de la generación de juegos de caracteres nuevos (para el que no guste de los actuales) y otro de la alteración de archivos que hayan sido creados incorrectamente.

Como conclusión, más que un añadido que pudiera interesarnos ponerle al QL, es una prolongación del propio QL que nunca debió ser amputada.



BCPL, LENGUAJE PARA CONNOISEURS

El BCPL, cuyas siglas corresponden al inglés BASIC COMPUTER PROGRAMMING LANGUAGE (Lenguaje Básico de Programación de Ordenadores), no es un lenguaje en absoluto similar al BASIC, nuestro viejo amigo, sino que se trata de una herramienta creada para el desarrollo de sistemas operativos, compiladores e intérpretes de otros lenguajes. Se engloba, pues, en la familia a la que pertenecen los lenguajes P (usado para la compilación de PASCAL) y C (usado en el desarrollo del sistema operativo UNIX), y es, de hecho, su "padre".

La implementación realizada en el QL se puede resumir en tres adjetivos: compacta, completa y elegante.

Se han incluido todos los comandos básicos del lenguaje tal y como se definen en su manual de referencia, incluyendo algunos que quedan obsoletos frente a comandos más completos creados específicamente para el QL, gracias a la gran flexibilidad del QDOS. Entre ellos cabe nombrar aquellos que se refieren al manejo de archivos y de pantalla, que se han reunido en uno solo con un parámetro que es el código de la operación a realizar, lo que evita la necesidad de realizar múltiples bifurcaciones tipo IF para acceder a estos recursos.

Uno de los detalles más interesantes que tiene esta realización del lenguaje es la posibilidad de unir rutinas creadas en código de máquina por medio de un ensamblador con rutinas creadas en BCPL en los puntos donde hiciera falta, y de permitir la creación de OVERLAYS, es decir, de fraccionar el programa de tal modo que se deje más memoria libre para datos, tomándose automáticamente de disco o mocrodrive las partes de programa que se necesiten y no se hallen en memoria en ese momento.

Sin embargo, se echa en falta algún programa de demostración complejo o alguna librería de rutinas interesantes ya grabados en el cartucho, aparte de la librería estándar de entrada/salida del BCPL.

MONITOR PARA EL QL

El monitor de código de máquina para el QL desarrollado por la casa Q-JUMP para Sinclair es de un tipo absolutamente clásico y, tal vez, excesivamente sencillo.

Al invocar al monitor por medio de QMON, aparece una línea encabezada por "Qmon> " en la que podemos escribir diversos comandos. Dichos comandos se pueden englobar en cuatro categorías: de ejecución, de visualización, de modificación y varios.

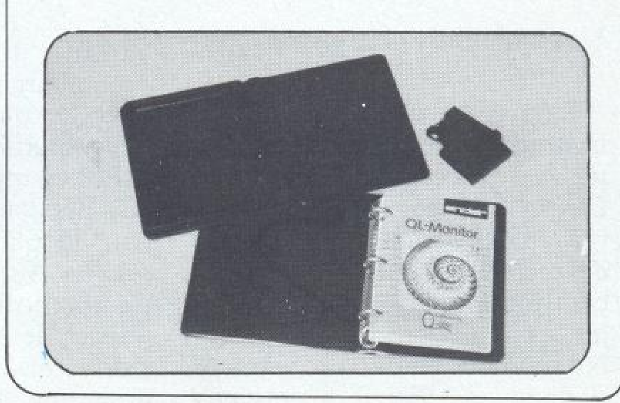
Los comandos de ejecución son los que tienen la función de supervisar el modo en que se ejecuta un programa o tarea. Tales son la ejecución paso a paso y la inserción de puntos de ruptura (BREAKPOINTS).

Los comandos de visualización muestran en pantalla los contenidos de la memoria o de los registros, o el listado desensamblado de las tareas.

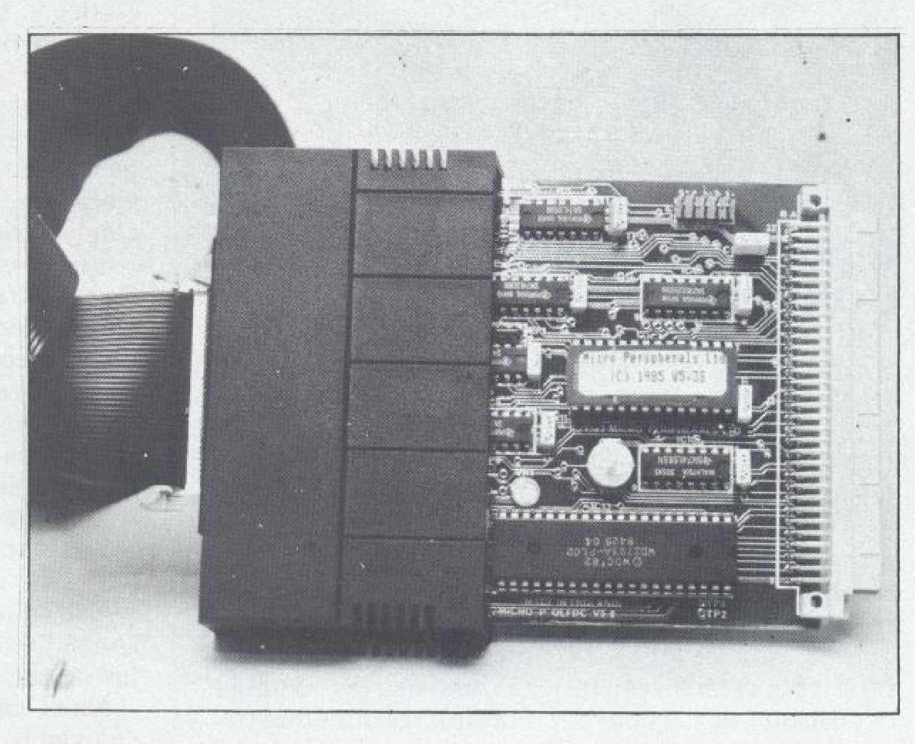
Los comandos de modificación permiten variar a voluntad los contenidos de los registros o de la memoria para ejercer un control extra sobre el funcionamiento de las tareas, o corregirlas sobre la marcha.

En el capítulo de varios se engloban dos comandos. Uno permite abrir canales para entrada/salida, para evitar tener que volver al BASIC para realizar estas tediosas labores. El otro, permite calcular los valores numéricos de direcciones como composición de los valores contenidos en diversos registros más unas cantidades fijas. Esto es muy útil en una máquina como el QL, en la que todas las tareas en código de máquina han de ser relocalizables y, por tanto, se basan en el direccionamiento relativo.

Como conclusión, este monitor es una herramienta básica en la depuración de programas de código de máquina, pero no pasa de eso.



Discos para el QL



En un ordenador con un procesador tan potente como el 68008, 128 K de memoria y unos programas de gestión excelentes, sólo queda el problema del almacenamiento masivo para que el QL sea una verdadera alternativa a los diversos PCs del mercado. La unidad de MicroPeripherals, comercializada en España por Investrónica, es la primera alternativa en nuestro país.

El sistema operativo del QL dedica buena parte de su código a conseguir un funcionamiento eficiente del *microdrive,* con excelentes resultados. La carga y grabación de programas son bastante rápidas, así como la copia de ficheros. Donde comienza a hacer agua el planteamiento es en el manejo de ficheros. Si bien el procesamiento secuencial de información no plantea grandes problemas, acceder a zonas separadas del fichero continuamente hace que los tiempos de espera aumenten, mientras la cinta se fatiga por el giro constante, planteando problemas de fiabilidad. La solución: añadirle una unidad de disco si queremos utilizar esta máquina en aplicaciones profesionales con acceso a muchos datos.

Un disco muy compacto

Para su capacidad de 720 K, la unidad tiene unas dimensiones verdaderamente pequeñas. El *interface* se introduce en el bus de expansión a la izquierda del teclado, y sobresale apenas 7 centímetros del aparato. Una larga cinta comunica al controlador con la unidad propiamente dicha, de 3" 1/2, 80 pistas y doble cara. A nueve sectores por pista, hace un total de 1.440 sectores (720 K). El controlador maneja hasta cuatro unidades de disco que funcionen con el estándar Shuggart, y tengan alimentación independiente. Por tanto, se pueden utilizar, si disponemos de ellas, viejas unidades de simple o doble cara, en cualquier formato. En cualquier caso, es necesario que sean de doble densidad.

El funcionamiento del *diskette* desde SuperBasic se realiza igual que todos los dispositivos de entrada/salida, referenciándose la nueva unidd como "fdk1_". El *interface* incluye, sin embargo, algunos comandos en ROM que facilitan el uso del nuevo periférico.

Por ejemplo, algunos programas pueden plantear problemas para adaptar su funcionamiento a disco. Además, al estar activos los microdrives y esta unidad, la reserva de memoria para tablas y directorios es mayor. Si algún programa no funciona correctamente en disco, siempre nos queda la opción de incluir el comando MSET, que hace que los *floppies* emulen al *microdrive:* cualquier referencia de nuestro programa a *microdrive* será llevada a cabo por el disco. Esta opción se puede manejar también por *hardware,* mediante un puente del *interface.* En este caso, al no inicializarse los microdrives, la cantidad de memoria ocupada es menor, y desaparecen los problemas que pueden plantear programas excepcionalmente grandes.

El comando MSET tiene su contrapartida en FSET y VSET. El primero sirve para restaurar el nombre del dispositivo a "fdkX_". VSET nos permite elegir el nombre, por ejemplo, "dskX_" o "flpX_". Así se pueden hacer compatibles programas diseñados para otros sistemas de disco.

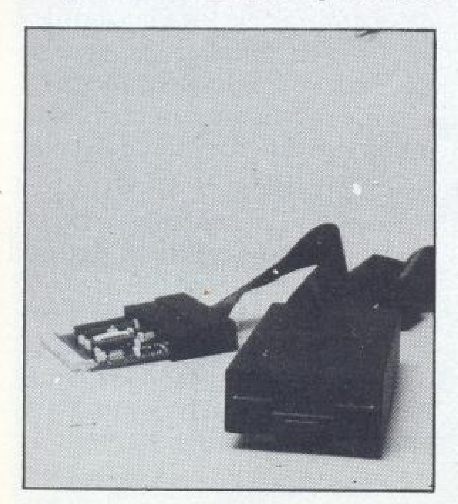
Otros comandos incluidos son SAVEO, SBYTESO y SEXECO. Los tres realizan la misma función que sus homónimos del SuperBasic, pero no dan

mensaje de error si el fichero existe. Permiten sobreescribir programas sin tener que usar el tedioso DELETE cada vez que deseamos actualizar un programa o fichero.

Para finalizar la lista de extensiones, los comandos DGET y DPUT proporcionan acceso directo a los sectores del disco. Con ellos se puede leer un sector cualquiera del disco a un vector entero de 512 elementos. El uso más obvio es para obtener réplicas de los discos sector a sector, opción conveniente siempre, y absolutamente necesaria para copias de seguridad en sistemas con un solo disco. También sirve para implementar un sistema de acceso aleatorio; para ello hay que saber cómo se organiza el directorio y los ficheros. Las herramientas para ello están, pero la implementación queda a cargo del usuario, lo que hace muy peligroso intentarlo, ya que una escritura en un sector incorrecto puede destruir todo el disco.

Programas de utilidad, para facilitar el manejo

La unidad se completa por un disco de utilidades, que incluyen un módulo ejecutable que acelera las copias entre disco y *microdrive,* ya que, aunque parezca paradójico, la copia de disco a microdrive es más lenta que entre dos cartuchos. Esto se debe a la gestión de bloques esclavos. El operativo utiliza toda la memoria sobrante para almacenar copias de sectores de *microdrives.* En caso de ser necesaria esta memoria, el operativo le indica al controlador que vuelva a escribir el bloque al cartucho, si es que había sido modificado.



Así, cuando copiamos un fichero de cartucho a cartucho, el operativo le busca un hueco en la memoria a la unidad de origen, y le da determinado tiempo para copiar todo lo que pueda de él. A continuación repite la operación en escritura, y es aquí donde se gana el tiempo, ya que cada sector que se escribe debe ser verificado. Para ello, se debe esperar a que la cinta dé una vuelta entera (unos 7 segundos). Al escribir varios sectores de una vez, el operativo no espera para cada sector, sino que hace las operaciones según pasa la cinta por la cabeza. Con el disco, éste devuelve sólo un sector cada vez, por lo que la espera se hace eterna si el fichero es largo. El programa que se proporciona hace las copias en bloques de 32 K, con una velocidd muy grande.

Otros programas permiten posicionar hasta nueve ventanas en cualquier lugar de la pantalla, cambiar una cadena de carácteres por otra en todo un fichero (muy útil para convertir programas de *microdrive* a disco, si se cambia mdv por fdk). DELETE_X sirve para borrados múltiples, y puede borrar hasta un disco (o *microdrive*) entero de una sola instrucción.

Otra de las utilidades permite redireccionar ficheros hacia otro periférico, sin interrumpir la ejecución de nuestro programa. Entre otras cosas, proporciona la posibilidad de un *spooler,* que manda los ficheros a impresora mientras hacemos otro trabajo.

Se completa la serie de programas con un editor de memoria o disco, que sirve para parchear *bytes* en los sectores de un disco o echarle una ojeada a la memoria.

Rápido, fiable y de gran capacidad

El nuevo sistema abre el camino a un uso profesional sin restricciones del QL: con una unidad simple se puede almacenar un libro entero, si trabajamos con QUILL; una gran cantidad de registros, si nuestro objetivo es ARCHIVE, y se mejoran los tiempos de acceso a la información. Para poder utilizar en disco los programas de Psion, es necesario ejecutar el programa config_bas, incluido en el cartucho de ABACUS, y que permite cambiar las unidades por defecto de los programas, las definiciones de impresora y la unidad donde se buscará el fichero de Ayuda.

Por lo demás, cualquiera que haya sufrido grandes esperar mientras se escribía el resultado de una compilación, o se ordenaba la base de datos, notará la diferencia de tiempos. Si el problema eran los malabarismos para conseguir meter un paquete en las 100 K de un cartucho, aquí hay 720. Si nos fatigaba el mensaje "medio incorrecto", los discos no suelen dar este tipo de problemas.

SOBRE EL SUPERBASIC

Cuando Sir Clive Sinclair denominó **SuperBasic** al lenguaje de programación que instaló en su QL, lo hizo por dos razones. La primera de ellas, de tipo comercial, surge de que cualquier producto que ponga la palabra SUPER en su propaganda suele vender más que uno que no lo haga. La segunda razón es más presuntuosa, ya que da a entener que dicho dialecto del **BASIC** es muy superior al lenguaje original y a todos los dialectos aparecidos hasta la fecha.

Una traición para arreglar una herejía

En la gran maraña de los lenguajes de programación de alto nivel, se distinguen dos tipos importantes: los lenguajes estructurados y los no estructurados. El representante más conocido del segundo tipo es el **BASIC.** En él existe una gran dependencia del programa con los números de línea en que van asignadas las sentencias. Esto es así por culpa de una carencia de estructuras de control de flujo, que se reducen a **GOTO, GOSUB, FOR...NEXT** e **IF...THEN,** lo que limita las posibilidades de programación.

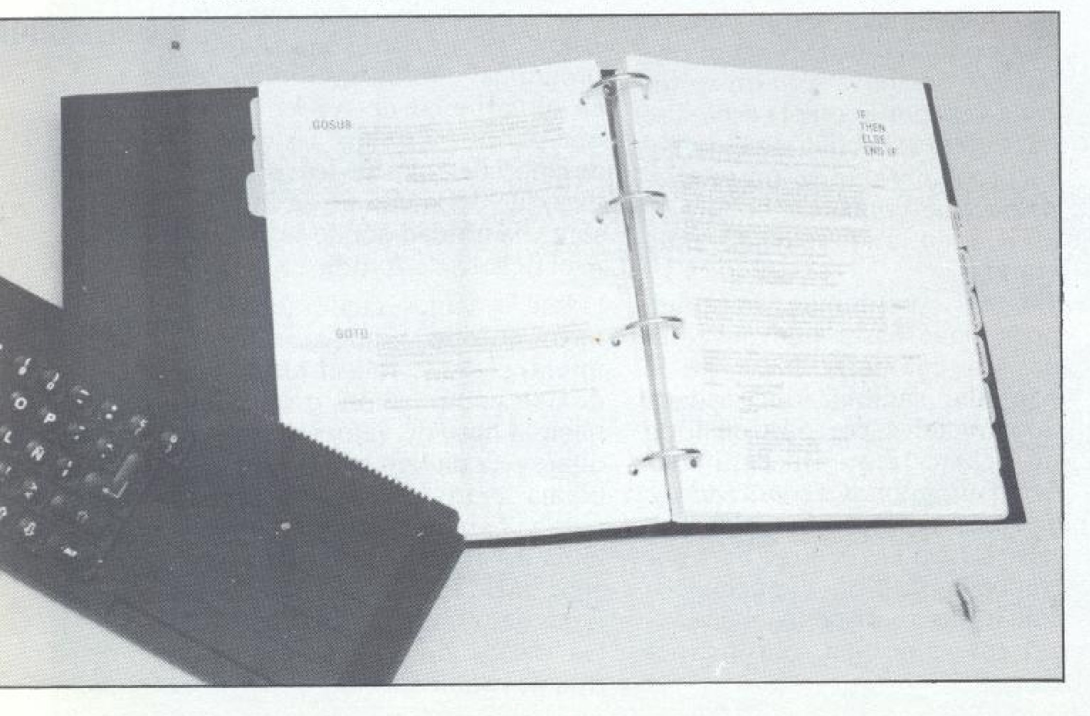
Por contra, los lenguajes estructurados, de los que el representante más conocido es el **PASCAL,** disponen de más formas de controlar el flujo del programa. Cabe destacar dentro de estas formas los *bucles* con salida condicional **(WHILE...DO, BEGIN...UNTIL),** la definición de secuencias que se pueden ejecutar llamándolas por su nombre **(PROCEDURE, FUNCTION)** y las decisiones múltiples **(CASE...OF).**

Los programadores profesionales suelen renegar de los lenguajes inestructurados, pues no permiten la creación de sistemas flexibles y fácilmente corregibles. De hecho, los consideran una herejía.

Pero los diseñadores del **BASIC** sabían perfectamente que es muy difícil para un novicio en los ordenadores aprender el manejo de las estructuras, y por eso hicieron un lenguaje más intuitivo, pero menos potente, puesto que los pensaban dedicar a la enseñanza.

Cuando Sinclair Reserch Ltd. presenta su dialecto, en el que se han flexibilizado las instrucciones **FOR...NEXT** y **IF...THEN,** se han añadido las definiciones de proceso **(PROCEDURE)** y función **(FUNCTION),** y se ha incluido la estructura **REPEAT...EXIST...END REPEAT,** lo que ha hecho es traicionar el espíritu de sencillez del **BASIC,** manteniendo y ampliando sus fines didácticos y deshaciendo a medias la herejía tradicional de este lenguaje. Porque, sin perder las estructuras clásicas del lenguaje, añade unas versiones sencillas de las estructuras propias de otros lenguajes que permiten dar el hasta ahora difícil salto entre ambos tipos de programación.

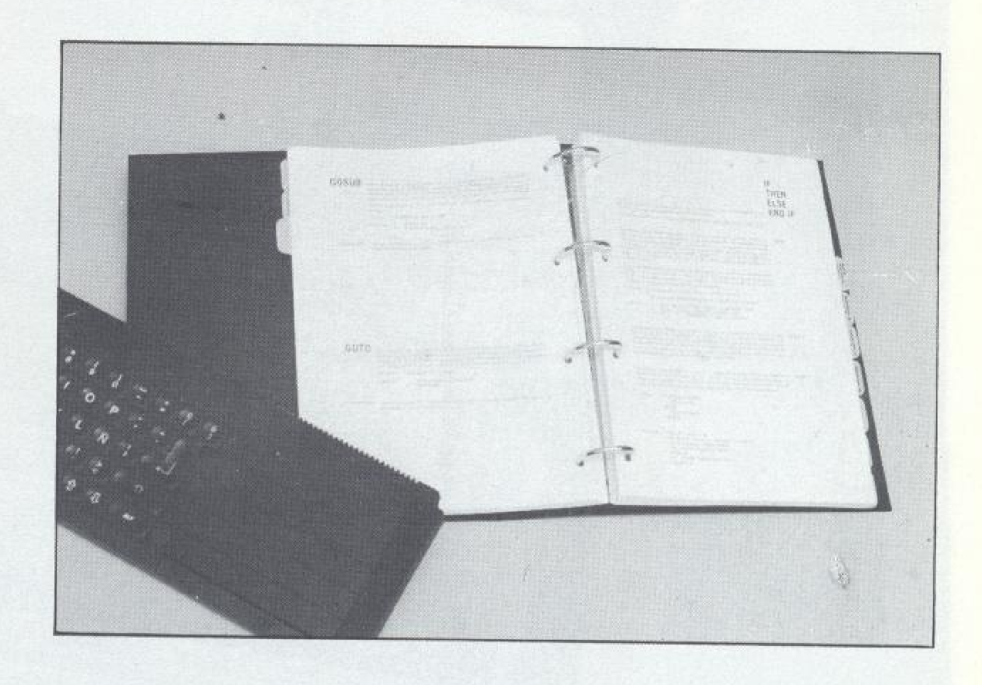
El manual del SuperBasic nos abre las puertas de un nuevo lenguaje.



Una bella labor de selección

Uno de los fallos que se puede encontrar en los viejos ordenadores de Sinclair, la familia ZX, es su completa desnudez en caso de producirse un error, que automáticamente devuelve el control al usuario.

En el QL, recogiendo la experiencia de otras marcas, por fin se ha incluido la posibilidad de manejar los errores desde dentro de los programas, sin que se haya de interrumpir necesariamente su funcionamiento. Para ello ha incluido dos estructuras y varias variables asociadas que,



como nota curiosa, hay que destacar que no están documentadas en ningún punto del manual.

Las estructuras son **WHEN EOF...END WHEN** y **WHEN ERR...END WHEN,** a las que se salta en caso de producirse un error por fin de fichero y un error cualquiera, respectivamente. Para poder identificar correctamente la causa del error, se han incluido también dos variables, **err_num** y **err_lin,** cuya función es conocer el código del error y la línea donde se ha producido, respectivamente.

Con esto, podemos programar más tranquilos porque podemos protegernos fácilmente de las típicas multiplicaciones de errores que se producían, por ejemplo, al darse un error y olvidársenos cerrar todos los canales que hubiera abierto el programa.

Un BASIC para manitas

Uno de los más importantes avances realizados con el **SuperBasic** es el simple modo en que se puede ampliar el conjunto de comandos a nuestra disposición. Aparte de la ampliación que puede representar la definición apropiada de procesos y funciones en **BASIC,** disponemos de la posibilidad de añadir procesos y funciones en código de máquina a través de una sola llamada al sistema operativo. A su vez, estos procesos y funciones de ampliación pueden hacer uso de los parámetros que se les pasen y del calculador de punto flotante con gran sencillez, de nuevo a través de unas cuantas llamadas al sistema operativo, todo lo cual está convenientemente documentado en varios libros al respecto. Esta propiedad es la que utilizan los dispositivos como discos y similares para incluir comandos específicos de dichos aparatos que permiten aprovecharse de sus especiales características. También se aprovechan de esto los llamados **"TOOLKITS",** conjuntos de instrucciones destinadas a hacer más fácil (o menos árida) la vida del programador. Una típica aplicación desarrollada en éstos, es el editor de pantalla completa, que permite moverse todo a lo largo de un programa EN SU LISTADO.

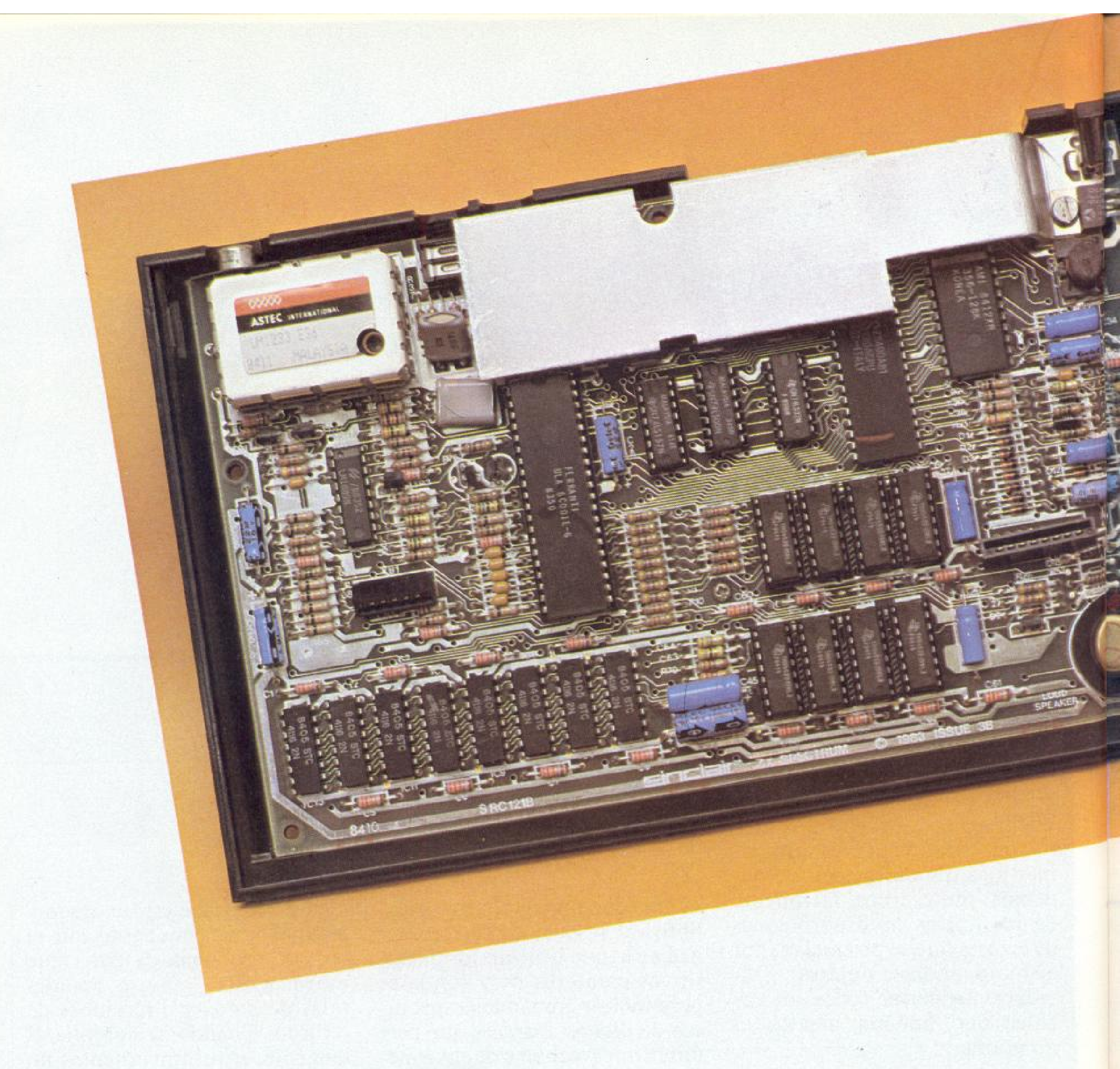
Los problemas del SuperBasic

Todas estas ventajas, siguiendo el viejo refrán "no hay mal que por bien no venga", habían de traer consigo una serie de inconvenientes nacidos de la misma naturaleza de dichas ventajas.

Tal es el caso de la velocidad del lenguaje. Cuando uno maneja un lenguaje estructurado y recursivo, se encuentra con el terrible problema de tener que mover arriba y abajo grandes pilas de datos y direcciones de retorno. Cuando se compila el lenguaje, estos movimientos no parecen lentos, pero al tratarse el **SuperBasic** de un lenguaje interpretado, la lentitud producida por ellos es agobiante. Esto no quiere decir que el lenguaje sea lento comparado con otros dialectos, pero sí que se podía haber conseguido que fuera más rápido. En cualquier caso, es mucho más rápido cuando maneja las estructuras que cuando interpreta los **GOTO** y **GOSUB.**

Se trata, pues, de un lenguaje que resulta lento frente a lo que pudiera haberse esperado del procesador que lleva, pero resulta un gran avance en velocidad frente a otros ordenadores de precio, que no características, comparable. Y, especialmente, es un "slato cuántico" **(QL, QUATUM LEAP)** frente a los anteriores aparatos de la casa Sinclair.





Introducción a la electrónica digital:

Puertas Lógicas



Un ordenador tiene una parte física ("hardware") y otra parte lógica ("software"), siendo ésta la más conocida. Por ello, vamos a ver la base del funcionamiento físico de los ordenadores. Y el elemento más sencillo que forma los "chips" del ordenador, la puerta lógica.

Todos sabemos que un ordenador es una máquina que puede efectuar ciertas operaciones. Pero no puede inventarse lo que ha de hacer, sino que hemos de decirle antes qué es lo que queremos que haga; o sea, programarlo. Vemos que hay dos problemas a resolver, ¿cómo almacena el ordenador sus programas? y ¿cómo trabaja realmente? Si le preguntamos esto a ese amigo que todos tenemos y que sabe tanto sobre ordenadores, nos dirá algo así como: "El ordenador almacena la información en sus circuitos de memoria, en forma de unos y ceros. Además, tiene otros circuitos que combinan esa información." Y, como casi siempre, nuestro amigo tiene razón, pero no hemos avanzado mucho.

Unos y ceros

De la respuesta anterior podríamos deducir que dentro de los circuitos del "Spectrum" vive una tribu de esforzados enanitos provistos de tizas y pizarras y que han sido entrenados en Sinclair para que guarden y trabajen con esa información. Bien, pues aunque parezca tonto, es parecido a eso pero sin enanos, así que no habrá que guardar el Spectrum en una jaula para evitar fugas de enanitos.

Lo cierto es que la información se guarda en forma de señales eléctricas, que son algo así como la tiza de los enanos, aquello con lo que el ordenador escribe. En cuanto a lo que escribe, el ordenador tiene su propio alfabeto, pero sólo con dos letras, "uno" y "cero". Ahora, la pregunta podría ser ¿por qué "unos" y "ceros" y no "unos", "doses" y "treses", por ejemplo? Bueno, no es un capricho de nadie. Simplemente, los dispositivos electrónicos que emplean estas máquinas trabajan sólo en dos estados, conducción y no conducción, también llamados encendido y apagado, abierto y cerrado, cortocircuito y circuito abierto, etcétera. Esta es la razón de que sólo sean dos las "letras" del alfabeto del ordenador. Y los nombres "uno" y "cero" es porque son los dos primeros números.

A la parte de la Electrónica que trabaja unos y ceros, es decir, con dos estados, se le denomina "Electrónica Digital", mientras que a la que trabaja sin esta limitación de funcionamiento en dos estados es la "Electrónica Analógica", y es la que trabaja con los equipos de sonido, las televisiones, etcétera.

Para construir cualquier dispositivo suelen hacer falta varias puertas. El número de ellas depende de la complejidad.

¿Uno o cinco?

Si alguna vez alguien se arma de las correspondientes herramientas y empieza a medir tensiones (voltios) en un circuito, utilizando para ello un voltímetro y mucho cuidado, verá que, en los puntos donde hay un "cero" hay 0 voltios, lo que resulta lógico. Pero que no se espere encontrar 1 voltio donde haya un "uno", porque lo más probable es que se encuentre 5 voltios. Esto es porque los dispositivos electrónicos lo pasarán bastante mal para distinguir entre 0 voltios y 1 voltio, pero se las arreglan muy bien para distinguir entre 0 y



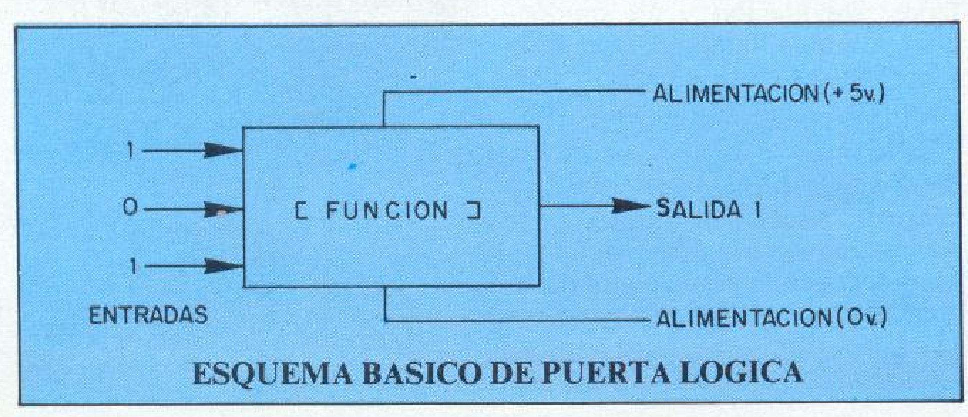
5. (A veces, en lugar de 5 voltios encontraremos 4 ó 6, pero nunca 1 ó 1,5. Y, en lugar de 0 voltios podrá haber 1 voltio, pero no 3 voltios). Habitualmente se asigna al cero una tensión de 0 voltios, y al uno se le asigna una de 5 voltios, aunque algunas veces pueda ser al revés. Normalmente, se indica la asignación.

Ya sabemos cuál es el aspecto que adquiere la información en el interior de un ordenador. Esa es la tiza, pero ¿cómo es la pizarra? ¿dónde se guarda? Sea lo que sea, debe ser algo en donde podamos dejar esa información todo el tiempo que sea preciso, pero también debemos poder leer o cambiar esa información en cualquier momento. Además, debe haber alguien que sume y reste.

Puertas lógicas y funciones lógicas

Ese algo son las puertas lógicas. Son capaces de transformar las señales eléctricas que reciben en otras, sin ayuda de elementos mecánicos, y sólo necesitan una alimentación eléctrica, que se soluciona conectándolo a una tensión de +5 voltios. Por supuesto, la combinación entre señales de entrada para dar una salida no es casual, ni varía, sino que es siempre igual. Es decir, que si cuando a la entrada hay dos "unos" y un "cero", a la salida aparece un "uno", bueno, pues cuando volvamos a tener esos mismos dos "unos" y el "cero", volveremos a tener un "uno" a la salida. De lo contrario, el desbarajuste sería impresionante y, por supuesto, los ordenadores no funcionarían (realmente, no existirían). En cuanto a las funciones lógicas, son la relación que existe entre las señales que entran y las que salen. Conociendo cuál es la función lógica de una puerta lógica podemos saber cuál será la salida para unas ciertas entradas y, el revés, qué entradas necesitamos para obtener una cierta salida.

Para ver con un poco más de profundidad qué es una función lógica, tendremos que pedir ayuda a las Matemáticas (pero muy poca, no hay que alarmarse). Hemos dicho que todos los ordenadores (también el Spectrum) trabajan con sólo dos clases de símbolos. Gracias a ello, y a algunas cosas más, podemos definir un Algebra de Boole (Boole fue un matemático inglés del siglo XIX), lo que quiere decir que se cumplen un montón de cosas útiles para nosotros. Exactamente es el Algebra de Boole de funciones lógicas, y es útil para nosotros ver que no sólo trabaja con los unos y ceros del ordenador, sino con muchas más cosas, algunas de ellas muy familiares para nosotros. Sobre todo, utilizaremos el lenguaje. Pero antes, veamos otras cosas.



ESQUEMA BASICO DE PUERTA LOGICA

Funciones AND y OR

Esas otras cosas van a ser las funciones lógicas más sencillas. Y, curiosamente, las tenemos todos los días cerca. Por ejemplo, una cadena de eslabones es un ejemplo de función lógica; exactamente, la función lógica AND. (En español es "función lógica Y"). Una cadena está formada por un eslabón y otro eslabón, otro eslabón y otro y otro, etcétera. Si falta algún eslabón, ya no tenemos cadena. Otro ejemplo. Supongamos que queremos ir de Villacolgada de arriba (pueblo A) a Aldeapeñas de abajo (pueblo B). Para poder recorrer el camino, hay que cruzar los puentes 1, 2 y 3. Es decir, el puente 1 "y" el puente 2 "y" el puente 3. (Figura 1).

Con uno solo de los puentes que estuviese derruido, nuestro camino quedaría interrumpido y habría que regresar al punto de partida sin poder realizar el viaje. La función lógica "Y" se comporta igual; basta con que "falle" uno de sus componentes para que "falle" toda la función.

Veamos ahora un par de ejemplos del funcionamiento de la función 0. Supongamos que consultamos el mapa de la figura para decidir el camino a seguir para viajar desde A hasta B. (Figura 2).

Tenemos tres caminos distintos, y, si suponemos que tardaremos lo mismo por cualquiera de los tres, nos resulta totalmente indiferente escoger cualquiera de ellos. Esto es, podemos ir por el 1 "0", por el 2 "0", por el 3. Si alguno de ellos estuviera cortado siempre podríamos utilizar cualquiera de los otros dos. Aquí aparece la característica de las funciones "0", y es que nos

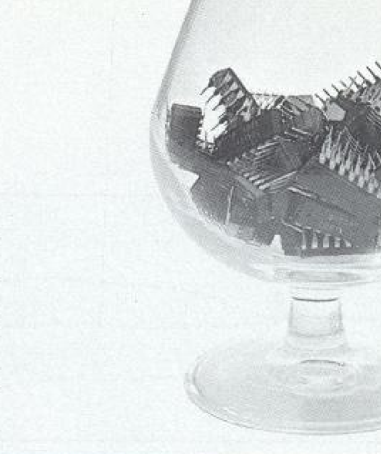
basta con que "funcione" uno de sus elementos.

Otro ejemplo de función "0" levemente distinta es el siguiente. Supongamos que tenemos un embudo y tres grifos. (Figura 3).

Para que salga agua por el embudo basta con que esté abierto el grifo 1 "0", el grifo 2 "0", el grifo 3, "0" dos de ellos "0" los tres a la vez. El único caso en el que no habrá agua en el embudo será cuando los tres grifos estén cerrados. La diferencia entre las dos funciones 0 es que en el primer caso (los caminos) podíamos escoger uno de los caminos, pero no todos, mientras que en el segundo los tres grifos pueden estar abiertos y tenemos un resultado "válido".

Curiosidades de la forma de hablar. Variables

Vamos a aprovecharnos del lenguaje para hacer experimentos sin romper ni estropear nada. Al hablar, combinamos palabras para crear frases, lo que no es ningún descubrimiento. Esas frases son de distintas clases, algunas son preguntas, otras, exclamaciones o comentarios sin demasiado sentido.



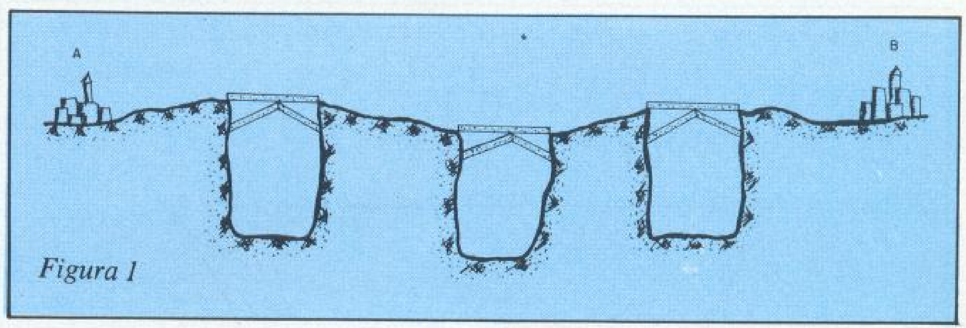


Figura 1

Pero hay un grupo de frases que nos interesa y que es el que vamos a utilizar, que son aquellas que utilizamos para decir algo en concreto, dar una opinión, transmitir una información, y afirmar o negar algo. Frases de esta clase son "el fuego está caliente", "en el Polo hace frío", "un kilobyte son 1.024 bytes", etcétera. Normalmente somos buenos chicos y decimos la verdad, es decir, sólo utilizamos frases que son ciertas. Pero, bueno, en el fondo somos humanos y, al-

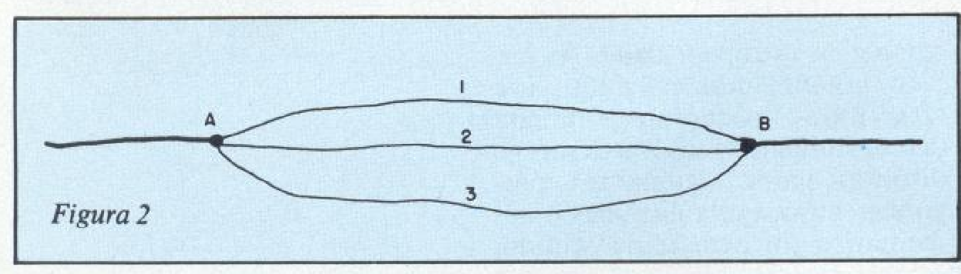


Figura 2

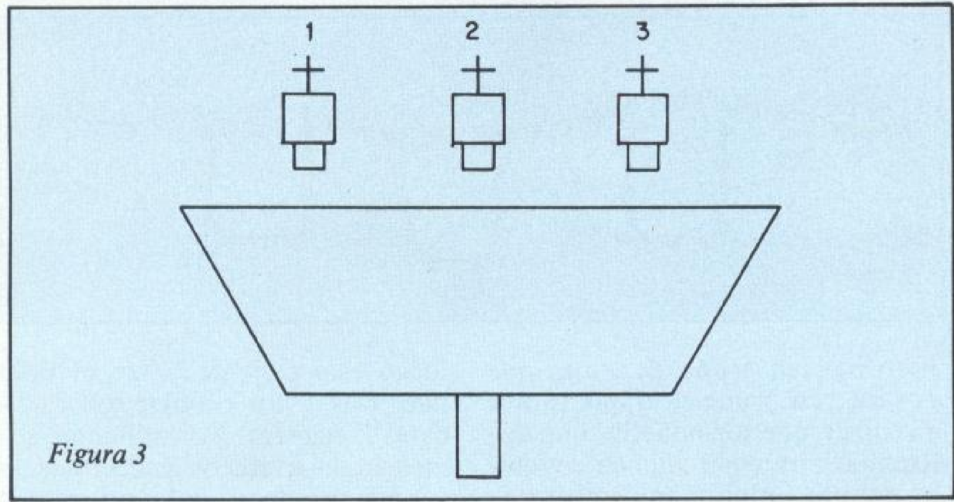
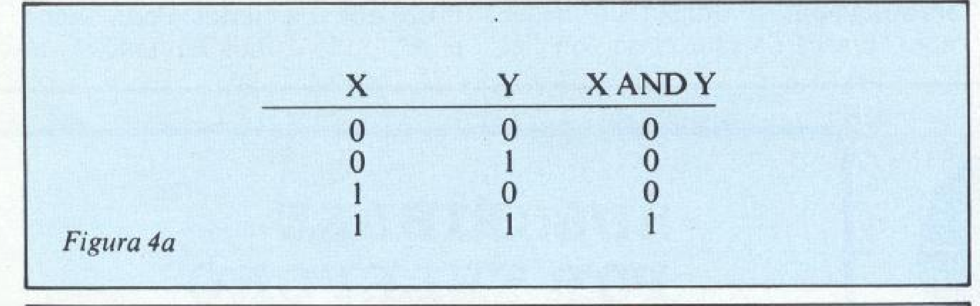


Figura 3



X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Figura 4a

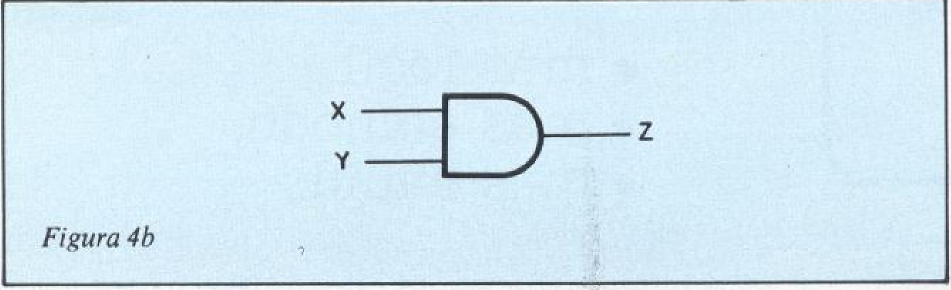


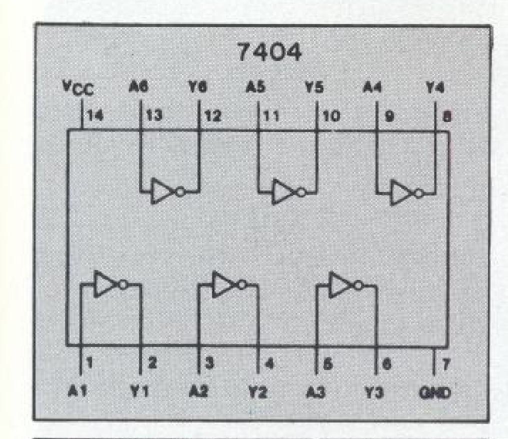
Figura 4b

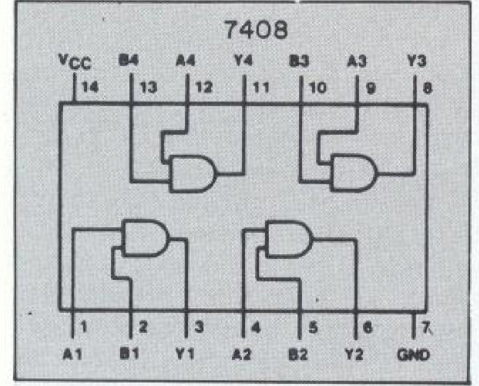
guien, alguna vez, dice alguna mentira, y para ello emplea frases falsas. Como vemos, las frases de esta clase sólo pueden ser verdaderas o falsas. De nuevo tenemos "algo" que está exclusivamente entre dos estados, "verdadero" o "falso". Podemos pensar (y de hecho es cierto) que las funciones lógicas funcionan igual de bien con frases verdaderas o falsas que con líneas eléctricas que tienen +5 voltios ó 0 voltios. Es más, si hemos dicho que los +5 voltios es un "uno" y los 0 voltios es un "cero", ¿por qué no asignamos el "uno" a las frases verdaderas y el "cero" a las falsas? Lo cierto es que no sólo podemos pensarlo, sino también hacerlo, y funciona bien. Es más, vamos a dar ya una definición: variable binaria es aquellos que puede valer uno o cero, ya sea una frase (verdadera o falsa), una línea del ordenador (+5 voltios ó 0 voltios), un interruptor (abierto o cerrado), un camino (que puede recorrerse o estar cortado), una puerta de una habitación (por la que se puede pasar o no), etcétera. Se les suele llamar x, y, z...

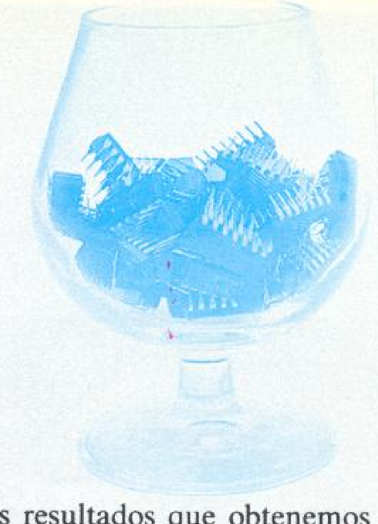
Funciones y puertas: "Y"

Ahora que ya disponemos de un buen montón de frases dispuestas a ser tratadas como pacientes y simpáticos conejillos de indias (y a los que no les va a doler nada), vamos a hacer pruebas. Dijimos que una frase era una combinación de palabras con sentido. Pero si unimos frases, ¿qué obtenemos? Hagamos la prueba con "Y". Por ejemplo, combinemos "hoy es día 30" con "hoy es lunes". El resultado será "hoy es día 30 Y hoy es lunes". Otro ejemplo, uniendo "mi coche es pequeño" con "el niño es alto" mediante "y", tenemos "mi coche es pequeño Y el niño es alto". No importa demasiado lo que se les pueda ocurrir a nuestros vecinos si nos oyeran utilizar frases como esta, puesto que nuestro problema es si la frase combinada es cierta o falsa. Es decir, ¿cuál es el resultado de combinar dos frases ciertas? ¿y dos falsas? La respuesta es... probémoslo.

Combinemos dos frases ciertas, por ejemplo, "el fuego es caliente" con "el hielo es frío". El resultado es "el fuego es caliente Y el hielo es frío", lo cual es cierto. Luego, la combinación de dos (o más) frases ciertas es cierta. Supongamos que las frases son "los pozos son profundos", que es cierta, y "las montañas son de cartón", que, salvo en decorados, es falsa. La combinación es "los pozos son profundos y las montañas son de cartón" y, desde luego, no es cierta. De aquí que baste con que una de las frases sea falsa para que lo sea la combinación (es algo así como con los puentes). Entonces, ¿qué es lo que hace una puerta "AND"? Pues simplemente, realiza la función lógica "Y" de las señales que tenga a la entrada, y aparece a la salida el resultado. Es decir, si todas las entradas son unos, la salida será uno.







Si tan sólo una de las entradas fuese cero, la salida sería cero.

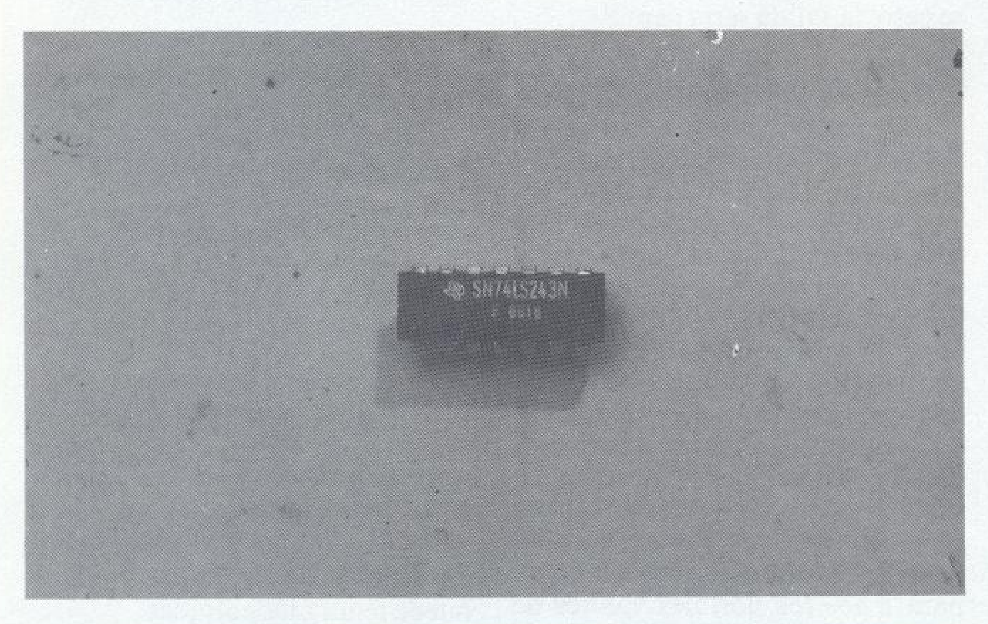
Tablas de verdad y símbolos

Para presentar con claridad el funcionamiento de una función lógica, utilizaremos las tablas de verdad. Su nombre viene de la época en que se aplicaban únicamente a las combinaciones de frases, para ver si la frase resultante era verdadera o falsa. En la figura 4a aparece la tabla de verdad de la función "Y", que utilizaremos como ejemplo. En primer lugar aparecen las variables que intervienen en la función. Como nos interesa saber cuál es el resultado de la función para *todos* los posibles valores de las variables, escribimos debajo todas las combinaciones posibles de valores que pueden tomar esas variables. Por último, escribimos los resultados. Un aspecto interesante: los resultados que obtenemos son los mismos que si multiplicamos los valores de X e Y: 0.0=0; 0.1=0; 1.0=0; 1.1=1. Por eso a veces puede escribirse en lugar de X AND Y, X.Y.

A la hora de hacer el esquema del circuito tenemos que representarlo de forma que resulte cómodo y que todo el mundo pueda entenderlo. El símbolo de la figura 4b es uno de los más usados; también puede tener más entradas.

Más funciones: "0"

También podemos utilizar "0" para combinar frases. Pero con



En este circuito integrado podemos ver las patillas y la numeración en la parte central.

```
  10 PRINT AT 1,3;"x";AT 1,5;"y"
;AT 1,7;"z"
  20 PRINT AT 1,9;"f";AT 1,11;"g
";AT 1,13;"h";AT 1,15;"FUNCION"
  30 LET linea=3
  40 FOR x=0 TO 1
  50 FOR y=0 TO 1
  60 FOR z=0 TO 1
  70 LET f=x OR y
  80 LET g=x OR z
  90 LET h=y OR z
 100 LET funcion=f AND g AND h
 110 PRINT AT linea,3;x;AT linea
,5;y;AT linea,7;z
 120 PRINT AT linea,9;f;AT linea
,11;g:AT linea,13;h;AT linea,17;
funcion
 130 LET linea=linea+2
 140 NEXT z
 150 NEXT y
 160 NEXT x
```

Figura 4

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

X
Y
Z

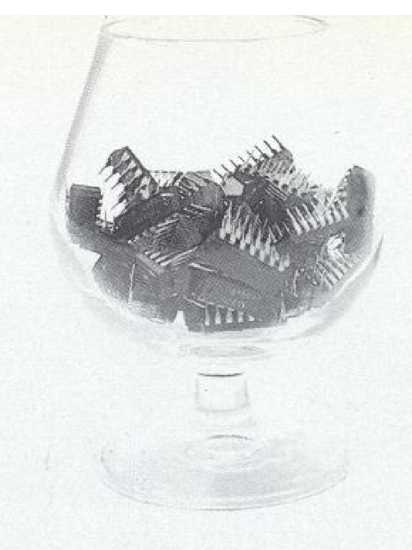
Figura 5

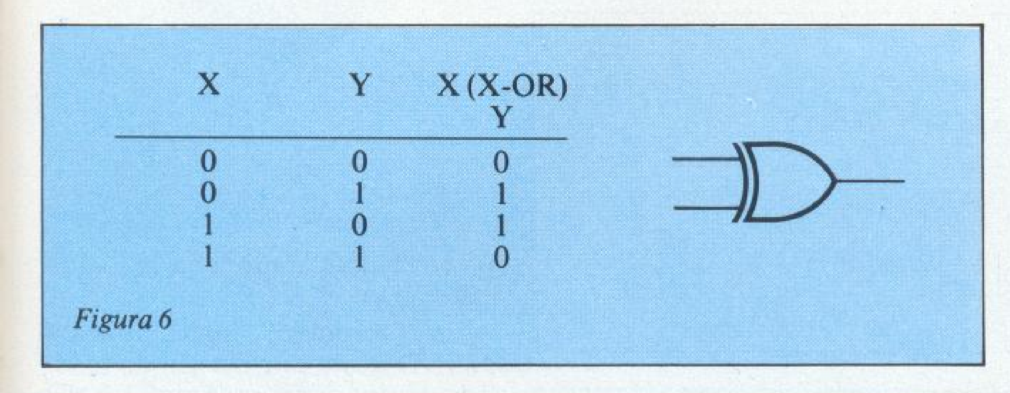
cuidado, porque puede ser de dos tipos: uno que une y otro que excluye. Así, el primer "0" es el de "iremos el jueves O el viernes". En principio, podemos ir el jueves, podemos ir el viernes o ir los dos días. El otro "0", llamado "0 exclusivo" o X-OR (de Exclusive OR) es el de "¿café o té?". Lo normal es tomar café, té, o ninguño, pero no (salvo maniáticos, excéntricos o especiales) los dos.

Veamos cuándo son verdaderos o falsos. El primer tipo será cierto cuando lo sea al menos una de las frases, así, si decimos "vendrá él o su hermano". Si aparece el individuo del que hablamos, la frase es cierta, aunque no venga su hermano (es decir, la segunda frase es falsa). Si sólo aparece el hermano, también hemos dicho la verdad aunque la primera frase haya sido falsa. Y, si aparecen los dos (las dos frases son ciertas), también es cierta. Resumiendo, basta con que una de las frases sea cierta para que lo sea la combinación, aunque pueden ser ciertas todas.

El comportamiento de la puerta lógica queda claro (Figura 5) y el símbolo que se utiliza para representarla en los circuitos es el que aparece al lado.

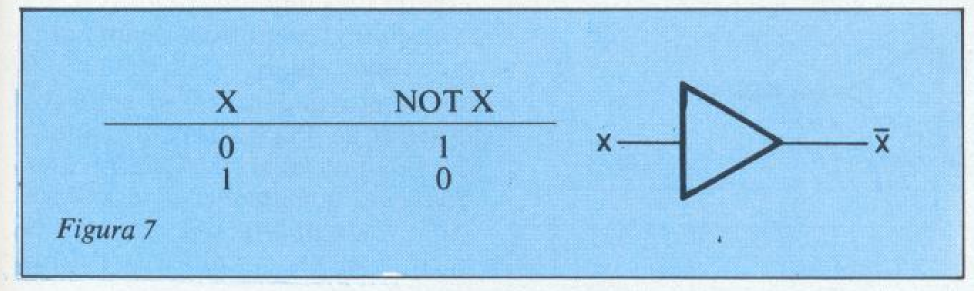
Pero hay otro tipo de 0; el exclusivo o disyuntivo. Es el de "hoy es lunes o martes". Si hoy es lunes y no martes, la frase es cierta (la primera es cierta y la segunda falsa). Si es martes (primera falsa y segunda cierta), entonces también es cierta. Si es miércoles, es falsa (las dos son falsas). Queda un caso, ¿qué ocurre si es lunes y martes? En tal caso, algo no iría nada bien, y, por lo menos, la primera conclusión que podemos extraer es que la frase es totalmente falsa. Luego, la combinación con señales eléctricas será la que aparece en la Figura 6, la salida será "uno" cuando una de las salidas sea





X	Y	X (X-OR) Y
0	0	0
0	1	1
1	0	1
1	1	0

Figura 6



X	NOT X
0	1
1	0

Figura 7

"uno", y "cero" cuando las dos sean "cero" o "uno" a la vez. El símbolo de circuito aparece junto a la tabla.

Es interesante observar aquí que la suma binaria y el X-OR son iguales. Así, 0 + 0 = 0; 0 + 1 = 1; 1 + 0 = 1; 1 + 1 = 1 0; (el cero es el resultado que vemos). Se usa el símbolo ⊕ para el "0 exclusivo" porque el "+" queda para la función "OR" vista antes.

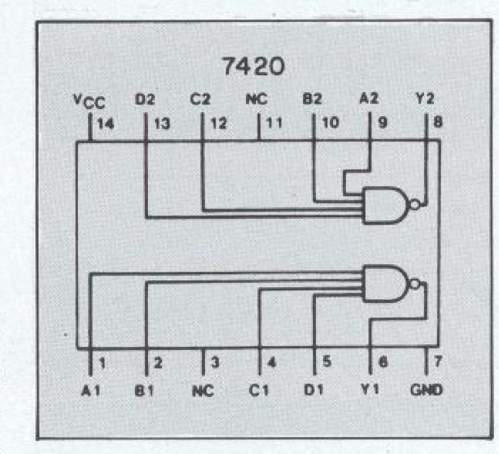
Todo al revés

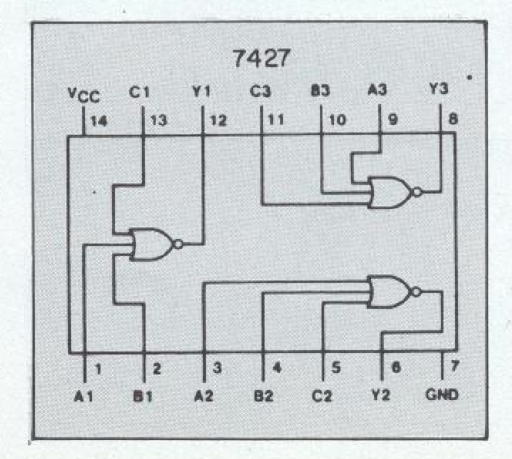
Falta una función básica: NOT o NO. Su actuación es cambiar todo: lo que es verdadero pasa a ser falso y lo que es falso pasa a ser verdadero. Es análoga a la palabra NO. Si a la frase "somos muy trabajadores", frase indudablemente cierta, le aplicamos el NO, queda "NO somos muy trabajadores", frase totalmente falsa. La particularidad de estas funciones es que sólo admiten una entrada. Así, las puertas lógicas que realizan la función "NO" sólo tienen una entrada. La tabla de verdad correspondiente aparece en la Figura 7, junto con el símbolo usado.

Generalmente, en lugar de decir NOT X se utiliza un guión, quedando $\overline{X}$

¿Para qué sirven?

Ya hemos visto qué son y qué hacen las puertas lógicas básicas. Ahora veremos algunos ejemplos sencillos. Así, si tenemos que detectar cuándo dos líneas de alarma





están a uno, pero sólo cuando estén a uno simultáneamente. Claramente, esa es la tarea para una puerta AND. Así conseguiremos tener la salida a uno cuando las entradas estén a nivel alto.

Si lo que nos interesase fuese saber si una (pero sólo una) de las dos estuviese a uno, nos bastaría con tener una puerta X-OR para tener la salida a uno. Si interesase detectar un uno o más, entonces sería preciso usar una puerta OR normal.

Aún hay más

Estas eran las puertas que realizan las funciones lógicas básicas. Pero hay otras que efectúan combinaciones de estas funciones básicas, así obtenemos las funciones NAND y NOR.

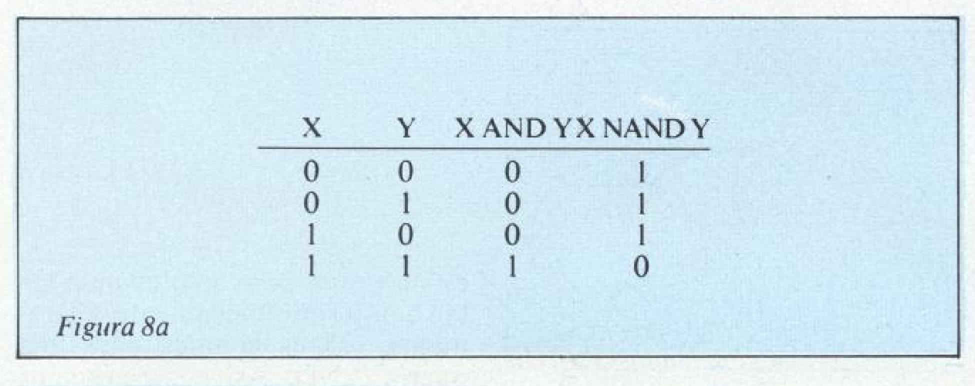
La función NAND (NO-Y) se consigue negando la salida de una función AND. La tabla de verdad que se obtiene aparece en la Figura 8a.

Para obtenerla, calculamos X AND Y y luego la negamos, para ello cambiamos los unos por ceros y los ceros por unos. El elemento de circuito será el que aparece en la Figura 8b, una puerta AND seguida de una puerta NOT.

Para simplificar, ponemos un círculo a la salida de la puerta AND, como se ve en la Figura 8c.

Por curiosidad, ¿qué función tendríamos si negamos las entradas de una función OR? Veamos su tabla (recordemos que X significa NOT X). (Figura 9).

¡Obtenemos de nuevo una función NAND! Luego, resulta que una puerta NAND puede ser también la mostrada en la Figura 10a.



X	Y	X AND Y	X NAND Y
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

Figura 8a

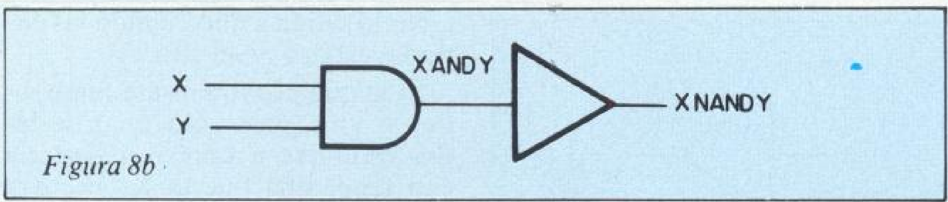


Figura 8b

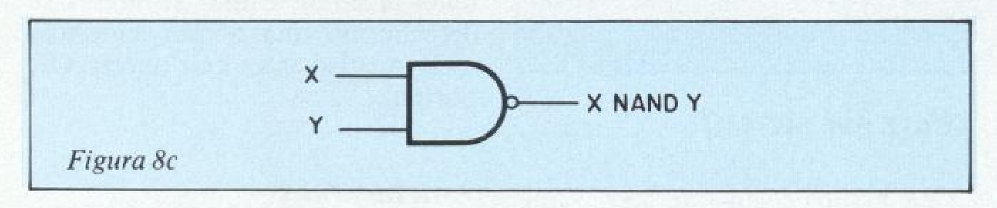
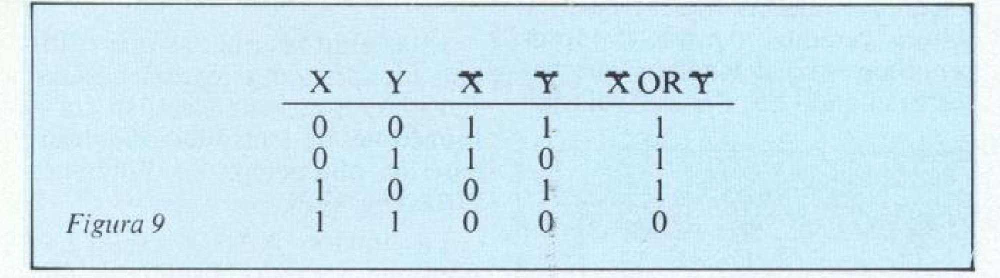


Figura 8c



X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X}$ OR $\overline{Y}$
0	0	1	1	1
0	1	1	0	1
1	0	0	1	1
1	1	0	0	0

Figura 9

X, Y, $\overline{X}$, $\overline{Y}$, X NAND Y

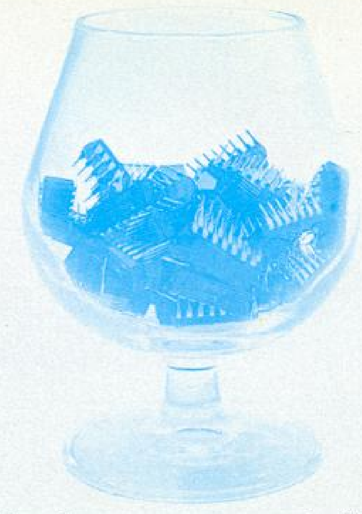
Figura 10a

X, Y, X NAND Y

Figura 10b

X	Y	X OR Y	X NOR Y
0	0	0	1
0	1	1	0
1	0	1	0
1	1	1	0

Figura 11



o bien la que aparece en la Figura 10b.

En resumen: una función NAND se obtiene o negando la salida de una función (puerta) AND o negando las entradas de una función (puerta) OR.

Veamos qué ocurre si negamos la salida de una puerta OR. Obtendremos una puerta NOR. El símbolo es el que aparece al lado de la tabla; una puerta OR negada, es decir, con un círculo a la salida. (Figura 11).

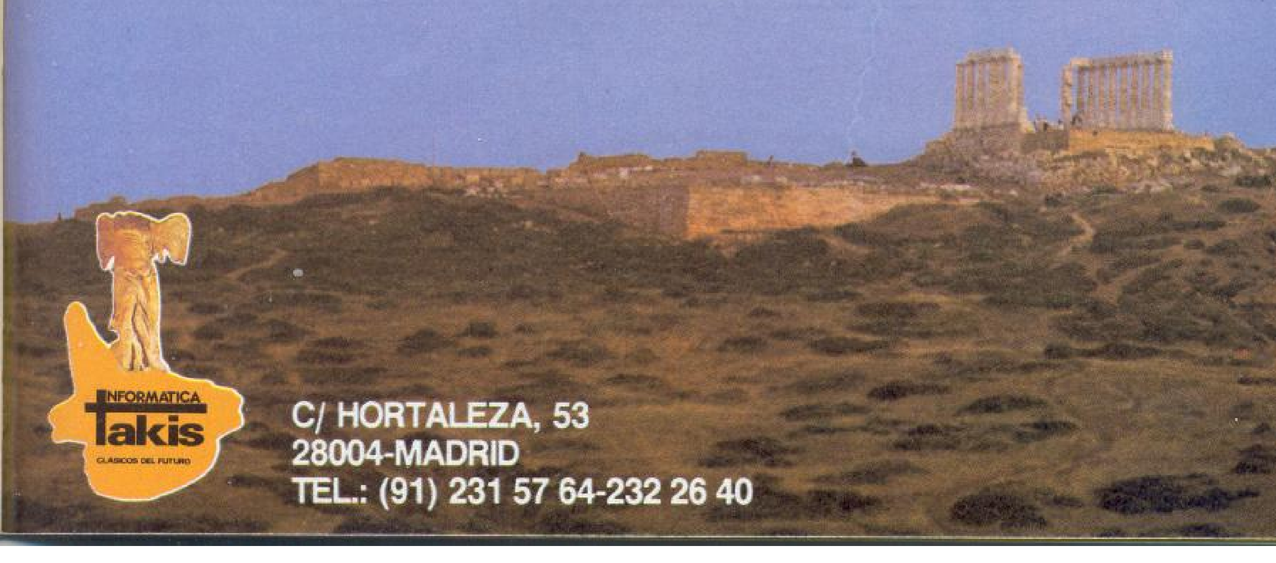
Y, lógicamente, también deberíamos llegar a una función NOR negando las entradas de una función AND, como se ve en la tabla de verdad de la Figura 12.

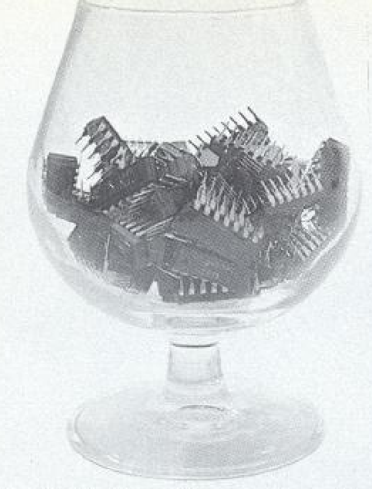
Por último, también podemos combinar puertas para obtener funciones más complejas. Para ello, hay que seguir ciertas normas. Si tenemos X AND Y OR Z, lo normal es evaluar la función de izquierda a derecha; es decir, primero calculamos X AND Y, y al resultado le aplicamos el "OR" con Z. De todas formas, para evitar ambigüedades, podemos emplear paréntesis; así, una expresión como (X AND Y) OR (Y OR NOT Z), puede calcularse por partes. Primero calcularemos NOT Z, después X AND Y e Y OR NOT Z. Por último, (X AND Y) OR (Y OR NOT Z). La tabla de verdad y el circuito están en la Figura 13.

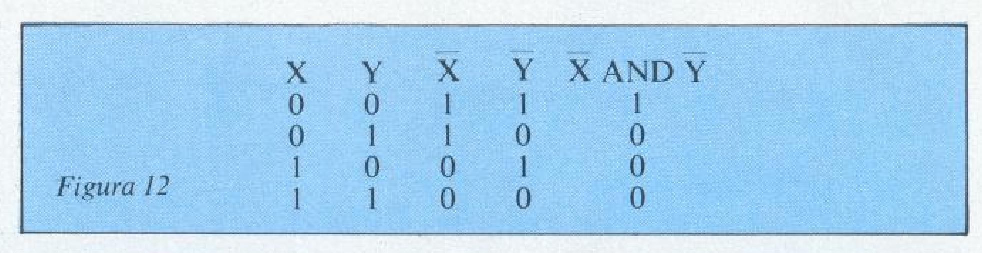
Y, ya casi para terminar, un pequeño programa para calcular funciones lógicas de tres variables con el "Spectrum". Cada variable requiere un ciclo FOR-NEXT (líneas 40, 50 y 60) y las funciones se calculan en las líneas 70, 80, 90 y 100. La función calculada es (X OR Y) AND (X OR Z) AND (Y OR Z) que se descompone en bloques; f es X OR Y; g es X OR Z y h



El ayer me ha creado,
hoy es el día de hoy
y yo soy el creador del mañana.







X	Y	$\overline{X}$	$\overline{Y}$	$\overline{X}$ AND $\overline{Y}$
0	0	1	1	1
0	1	1	0	0
1	0	0	1	0
1	1	0	0	0

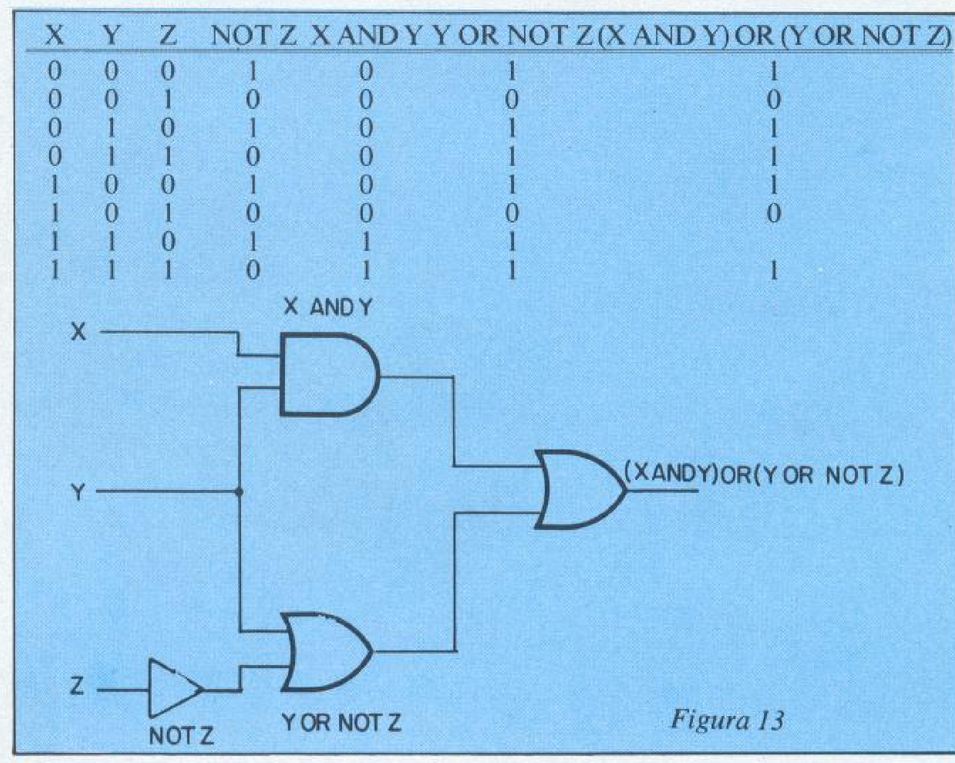
Figura 12

es Y OR Z. Función es el resultado total. Las otras funciones vistas, NOR y NAND se escriben como NOT OR y NOT AND (Figura 13).

En la tienda

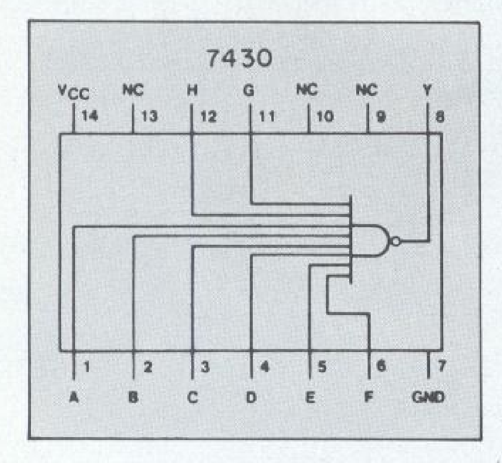
A continuación damos unos datos de interés a la hora de comprar y utilizar las puertas lógicas.

Las puertas lógicas se venden en circuitos integrados, conocidos por su número, que comienza siempre por 74. Después del 74 pueden ir una o varias letras, la "L" si la puerta es de bajo consumo, la "S" si es rápida, más rápida que las



X	Y	Z	NOT Z	X AND Y	Y OR NOT Z	(X AND Y) OR (Y OR NOT Z)
0	0	0	1	0	1	1
0	0	1	0	0	0	0
0	1	0	1	0	1	1
0	1	1	0	0	1	1
1	0	0	1	0	1	1
1	0	1	0	0	0	0
1	1	0	1	1	1	
1	1	1	0	1	1	1

Figura 13



normales, y "LS" si une ambas características. Algunas de las más típicas son:

7400: cuatro puertas NAND, con dos entradas por puerta.

7404: seis puertas NOT.

7408: cuatro puertas AND, con dos entradas por puerta.

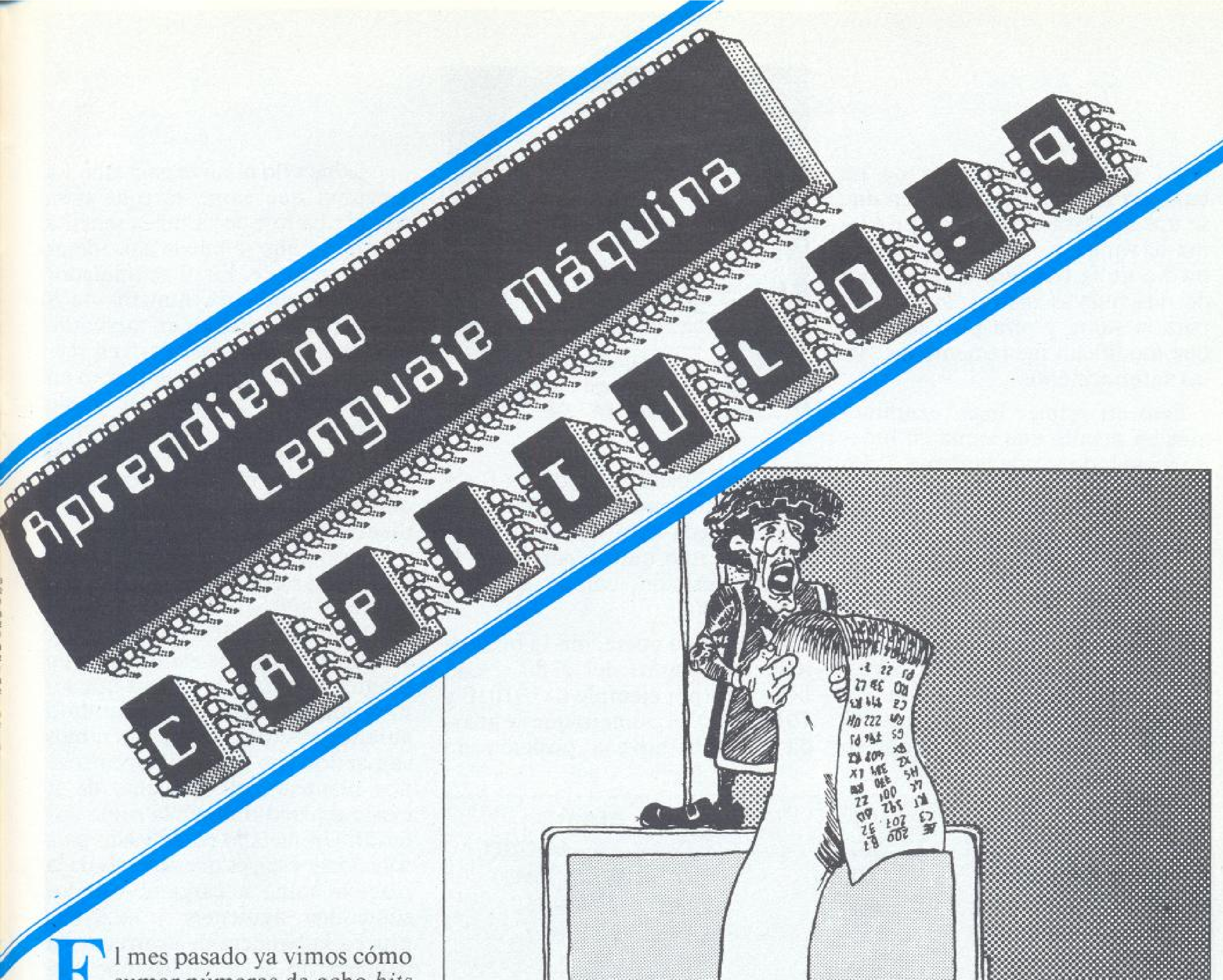
7420: dos puertas NAND con cuatro entradas por puerta.

7427: tres puertas NOR de tres entradas.

7430: una puerta NAND con ocho entradas.

Francisco López Larrio





El mes pasado ya vimos cómo sumar números de ocho *bits* utilizando diversos sistemas de direccionamiento y comparando las ventajas que ofrecían unos y otros. Este mes vamos a ver cómo restar dos números, también de ocho *bits*, y cómo hacer que estas operaciones se hagan con números mayores por medio de la bandera de acarreo.

Aprendiendo a restar

La operación de resta tiene un formato muy similar al mostrado el mes pasado con la suma pero con algunas restricciones.

Recordará que al realizar la operación ADD (suma), se debían indicar los dos sumandos, debiendo ser el primero un registro interno del Z-80, ya que en él se almacenaba el resultado. Esto proporciona gran flexibilidad a los programas al poder usar diversos registros, cada uno para una función. Con la operación de resta desaparece esta libertad y el primer registro (del que se resta) siempre es el A. Por ello el formato de la instrucción se simplifica ya que este primer operando no se indica y queda: «SUB xx», donde xx indica el número a restar, que puede indicarse con algunos de los diversos modos de direccionamiento que vimos en el capítulo anterior. Estos son los que se indican a continuación: Inmediato (el número a restar va a continuación de la instrucción), suma de dos registros (el A y cualquier otro), Indirecto con HL (HL indica la dirección de donde se saca el número) e Indexado con IX o IY (el número se saca de la dirección de memoria indicada por este registro más un desplazamiento que se da a continuación de la instrucción). Por lo demás el funcionamiento es igual a «ADD» con la excepción lógica de que en lugar de sumar se resta. Como ejemplo se da en la figura 1 el listado correspondiente a una resta de un *byte* por medio del direccionamiento indexado. Los otros tipos de direccionamiento se realizan igual que con la suma.

Múltiple precisión

Lo visto hasta ahora resulta interesante a nivel educativo pero muy poco útil en la práctica. En primer lugar está el problema de que sólo podemos operar con números de hasta 255 (FFh), con lo que al cal-

cular una sencilla cuenta de un bar, con los precios de hoy en día, se nos quedará inútil. Este problema le vamos a resolver ahora por medio de la bandera de acarreo y de dos nuevas instrucciones, una para la suma y otra para la resta, que modifican ligeramente las vistas anteriormente.

Pero en primer lugar examinemos en detalle una suma en binario ayudándonos de la figura 2. En esta se puede ver que se ha realizado una suma a mano del modo en que habitualmente la hacemos los humanos y siguiendo las reglas de la aritmética binaria, es decir, 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 0 y 1 + 1 = 10. Por lo que al sumar 0110 y 1011 decimos (empezando por la derecha como es habitual) 0 + 1 es 1, 1 + 1 es 0 (en esa posición) y nos llevamos 1 a la siguiente, con lo que en



la tercera posición queda 1 + 0 + 1 = 0 y nos llevamos 1 a la cuarta posición, quedando 1 + 0 + 1 = 0 y nos llevamos uno que se coloca en la quinta posición, con lo que nos queda 10001.

Pero si esta operación se hubiese realizado dentro del Z-80 y con ocho bits (por ejemplo 01101010 y 10110011), el número que se guardaría en la novena posición no puede hacerlo al no existir esta. La pregunta que surge es ¿que pasa con él? La respuesta no es sencilla de dar, ya que se puede considerar que desaparece. En el acumulador sigue habiendo un número de 8 bits y si lo guardamos en memoria, sólo guarda este *byte*. Pero en realidad si se ha almacenado, pero en la bandera de acarreo. De modo que en las operaciones aritmética se puede considerar esta bandera como un noveno *bit* del resultado, que almacena el 0 ó 1 correspondiente.

Supongamos ahora que queremos sumar dos números de dieciséis bits (00101101 10101101 y 10010001 11101100 por ejemplo). La suma de los *bytes* inferiores no plantea nigún problema ya que vimos como hacerlo en el capítulo anterior, pero cuando queramos sumar los dos *bytes* superiores se nos planteará el problema de si existe acarreo o no de la suma anterior. Un método complicado para solucionar esto, es que después de la primera suma se carga uno de los sumandos siguientes y se comprueba la bandera de acarreo (esta bandera no se modifica por una operación LD), si esta puesta, le sumamos uno al sumando y luego le sumamos el otro. Si no está puesta, sumamos el otro directamente. De este modo se sumaría el acarreo anterior si fuese un 1 (y puede considerarse que lo suma también si es un cero, ya que esto no afecta al resultado), pero de un modo complejo que nos haría escribir más instrucciones y enlentecería el programa.

Por fortuna para nosotros, existé una instrucción que hace todo esto automáticamente. Se llama ADC y funciona del mismo modo que la suma pero con la excepción de que al *bit* menos significativo (el de la derecha) del acumulador le suma también el contenido de la bandera de acarreo. Esto, como puede verse en la figura 3, resulta el equivalente del 'y me llevo 1' que hacíamos antes y nos permite extender la suma todos los *bytes* que queramos, bastando con que la instrucción que usemos sea ADC y obteniendo el resultado agrupado por *bytes*.

```
EA60                  10     ORG   60000
EA60  DD2154EC        20     LD    IX,60500
EA64  DD7E00          30     LD    A,(IX+0)
EA67  DD9601          40     SUB   (IX+1)
EA6A  DD7702          50     LD    (IX+2),A
EA6D  C9              60     RET
EA6E                  70     END
```

FIGURA 1

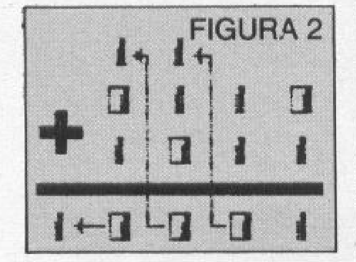


FIGURA 2

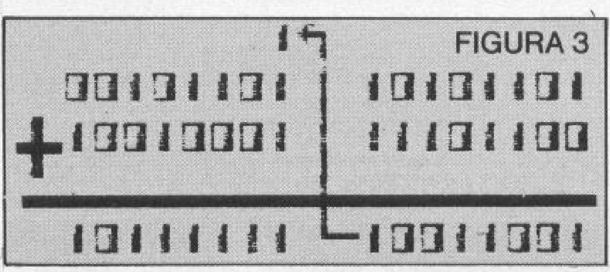


FIGURA 3

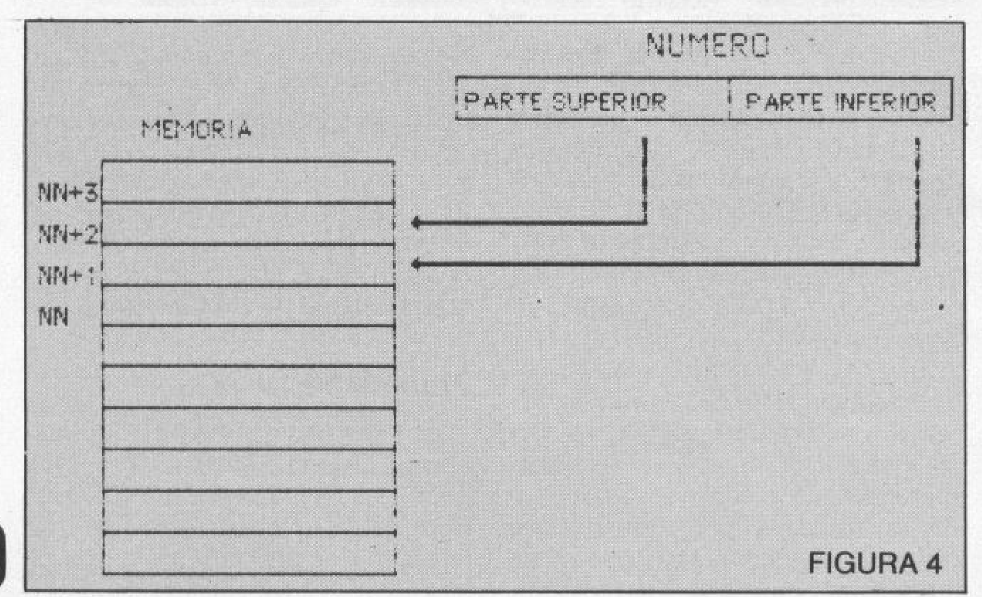


FIGURA 4

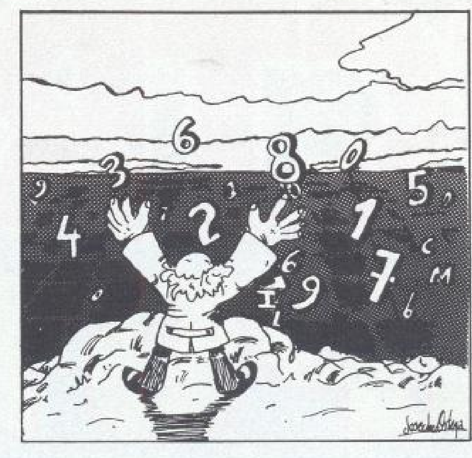


N.º 4
PRIMERA REVISTA EN CASSETTE
SPECTRUM
COMPUTING
PARA 16K Y 48K
3 D
Añada una nueva dimensión a su SPECTRUM.
Acción en tres dimensiones.
Busque y destruya la flota de ataque estelar.
Piérdase en nuestro laberinto en tres dimensiones en código máquina.
Defensa
Su oportunidad para venir en defensa del planeta.
875 Ptas.
Más sobre la programación de dibujos animados.
Clocks up
¿Sabe generar un reloj digital en su SPECTRUM?
Y MUCHO MAS
BIENVENIDO A
SPECTRUM
COMPUTING
LABERINTO EN TRES DIMENSIONES
GRAFICOS
WIZARD
RELOJ
DEFENSA
DIBUJOS ANIMADOS
ATAQUE ESTELAR
AGENDA TELEFONICA
SKI
PUZLE
LA SERPIENTE
MAS DE 150.000 PTAS. EN PREMIOS. BASES EN EL INTERIOR
Solicítela a: INFODIS C/ Bravo Murillo, 377 - 5.º A - 28020 MADRID
CUPON DE PEDIDO · Nº 4
Sí, envíenme al precio de 875 ptas.... ejs. de SPECTRUM COMPUTING
El importe lo abonaré: Contra reemboso □ Con mi tarjeta de crédito □
Adjunto cheque □ American Express □ Visa □ Interbank □
Número de mi tarjeta
Fecha de caducidad
NOMBRE
DIRECCION
CIUDAD
PROVINCIA
Sin gastos de envío

1Un programa de suma en 16 bits

Vamos a desarrollar esto que hemos explicado mediante la realización de un programa que sume números de 16 *bits* almacenados en posiciones de memoria consecutivas y guarde el resultado en otras dos posiciones de memoria.

Antes de empezar a hacer el programa debemos establecer cómo se va a almacenar la información. En este caso diremos que el primer número se almacena en las direcciones 60500 y 60501, el segundo en 60502 y 60503 y el resultado se almacena en 60504 y 60505. Pero con esto no queda especificado todo, ya que el número se puede



almacenar de dos formas distintas, según se introduzcan los dos *bytes* del número en la memoria. Pudiendo ser, como se ve en la figura 4, el *byte* inferior en la primera memoria y el *byte* superior en la segunda. El segundo formato posible representado en la figura 5 es el inverso del anterior. Almacenando el *byte* inferior en la segunda memoria y el superior en la primera.

Aunque el primero puede parecer el más lógico desde nuestro punto de vista, vamos a usar el segundo porque es el formato que usa internamente el Z-80 cuando tiene que almacenar números de dieciséis *bits* en memoria. Por lo que la distribución queda como se indica en la figura 6.

El sistema de direccionamiento que usaremos es el indexado por medio del registro IX, ya que es uno de los más versátiles. (No obstante se podría usar cualquier otro realizando las modificaciones oportunas.) Por ello lo primero que tenemos que hacer es cargar en IX la dirección que tomaremos de base para leer las memorias usando un desplazamiento. Esta dirección puede ser cualquiera lo suficientemente cercana (a menos de 127 de distancia) al rango 60500-60505. Para facilitar nuestra labor haremos que IX contenga 60500. Una vez hecho esto realizamos la primera suma como vimos anteriormente, es decir, cargando el acumulador con el primer *byte* del primer número por medio de "LD A,(IX+1)" (es la dirección 60501) y le sumamos el primer

FIGURA 5

NUMERO
PARTE SUPERIOR | PARTE INFERIOR
MEMORIA
NN+3
NN+2
NN+1
NN
FIGURA 5

FIGURA 6

MEMORIA	
60505	BYTE BAJO RESULTADO
60504	BYTE ALTO RESULTADO
60503	BYTE BAJO SUM. 2º
60502	BYTE ALTO SUM. 2º
60501	BYTE BAJO SUM. 1º
60500	BYTE ALTO SUM. 1º

Ensambladores: DEVPAC

El primer ensamblador que comentamos en nuestras páginas es, sin duda, uno de los más potentes y el más famoso, habiendo llegado a constituir un estándar con respecto al cual se comparaban todos los demás.

En realidad el DEVPAC es mucho más que un ensamblador, ya que dentro del mismo paquete se incluyen dos programas distintos: El primero, denominado GENS, es el correspondiente al ensamblador propiamente dicho y el segundo, MONS, es un potente desensamblador que nos permite examinar cualquier programa en lenguaje máquina.

En este caso nuestro interés se centra en el ensamblador y en el editor asociado que nos permite introducir el texto fuente para que posteriormente sea pasado a código máquina, ya que sin él no tendría razón de ser el ensamblador.

El editor

Este está orientado a líneas, es decir, al contrario de los programas orientados a caracteres como son los procesadores de textos, que nos permiten desplazarnos libremente por la pantalla para hacer modificaciones en cualquier punto; en los de líneas hay que modificar una línea de cada vez coartando bastante la libertad de acción. Un editor de este tipo que todos hemos usado es el que incorpora el BASIC y que nos permite modificar una sola línea de programa (aunque en pantalla ocupe más) y para desplazarse a otra hemos de terminar con la que estemos en ese momento.

Dentro de este modo de funcionamiento genérico el GENS tiene un grupo bastante amplio de comandos que hacen que sea mucho más potente que el del BASIC, pero también de más difícil manejo. Así, por ejemplo, para introducir líneas no se puede teclear el texto directamente, sino que previamente hay que decirle que entre en modo de inserción por medio de la orden "I". Otro ejemplo es cuando se edita una línea, en este caso en lugar de desplazarse libremente con las flechas del cursor y modificar introduciendo los caracteres y con el DELETE, hay que

byte del segundo, que está en 60503 por medio de "ADD A,(IX+3)". En este caso no se utiliza la suma con acarreo, ya que es el byte más inferior y no tenemos que añadir nada de ningún número anterior, y el contenido de la bandera de acarreo será el resultado de alguna operación anterior realizada por el microprocesador y que no nos interesa.

Para guardar este resultado parcial hacemos "LD (IX+5),A". Una vez hecho esto debemos sumar el *byte* superior de un modo muy similar al anterior, pero poniendo ADC en lugar de ADD, con lo que nos queda "LD A,(IX+0)" para cargar el segundo *byte* del primer sumando y añadiéndole el otro por medio de "ADC A,(IX+2)", con lo que se suman ambos y el acarreo anterior. Para almacenar esta otra parte del resultado hacemos "LD (IX+4),A", después de lo cual podemos devolver el control al BASIC por medio del RET. El listado completo se da en la figura 7.

Una vez introducido y ensamblado el programa podemos ejecutarlo usando el RANDOMIZE USR 60000, pero previamente han de introducirse los datos en las memorias que han de contener los sumandos. Esta vez es algo más complicado, ya que hay que dividir el número (comprendido entre 0 y 65535 que es la máxima capacidad de 16 *bits*) en dos *bytes* y luego sacar el resultado también como un número de 16 *bits*.

FIGURA 7

```
EA60                10      ORG  60000
EA60 DD2154EC       20      LD   IX,60500
EA64 DD7E01         30      LD   A,(IX+1)
EA67 DD8603         40      ADD  A,(IX+3)
EA6A DD7705         50      LD   (IX+5),A
EA6D DD7E00         60      LD   A,(IX+0)
EA70 DD8E02         70      ADC  A,(IX+2)
EA73 DD7704         80      LD   (IX+4),A
EA76 C9             90      RET
EA77               100      END
```

El primer proceso se puede hacer con la siguiente línea en BASIC

10 LET SUP=INT(NUM/256): LET INF=NUM−SUP*256: STOP

En la que se supone que el número está en la variable NUM y devuelve el *byte* inferior en INF y el superior en SUP, con lo que bastará hacer POKE dirección del inferior, INF y POKE dirección del superior, SUP.

El proceso inverso resulta más fácil, y basta con

PRINT PEEK(dirección del inferior) + PEEK(dirección del superior)* 256

para ver el resultado.

FIGURA 8

```
EA60                10      ORG  60000
EA60 DD2154EC       20      LD   IX,60500
EA64 DD7E01         30      LD   A,(IX+1)
EA67 DD9603         40      SUB  (IX+3)
EA6A DD7705         50      LD   (IX+5),A
EA6D DD7E00         60      LD   A,(IX+0)
EA70 DD9E02         70      SBC  A,(IX+2)
EA73 DD7704         80      LD   (IX+4),A
EA76 C9             90      RET
EA77               100      END
```

Resta con múltiple precisión

Al igual que la operación ADC de la suma, hay otra (SBC) que resta con acarreo y que nos permite modificar el listado anterior y convertirlo en uno que reste. Curiosamente esta operación puede realizarse con otros registros aparte de con el A (al contrario que la suma normal), por lo que se deben indicar los dos a restar. Siendo en nuestro caso el primero el A.

El SBC funciona igual que el SUB, con la excepción de que la bandera de acarreo se RESTA del resultado de la resta de los dos *bytes,* es algo similar a lo que nos sucede si hacemos 43 − 25, decimos 3 menos 5 es 8 y nos llevamos 1 y a continuación hacemos 4 menos 2 es 2 y menos una que nos habíamos llevado da 1, por lo que obtenemos 18.

Una vez explicado esto la modificación no plantea ningún problema y da el listado de la figura 8.

---

hacer uso de un repertorio amplio de instrucciones con las que se le indica que se desplace al principio o fin, que borre caracteres, que inserte, etcétera.

Por otra parte y para compensar esta complejidad, existe un conjunto de comandos que facilitan la edición sobre todo en programas largos. Un ejemplo es la "F", que realiza una búsqueda automática en un rango de líneas (o en todo el texto) hasta encontrar una cadena de caracteres específica. Otro es la "M", que copia el contenido de una línea en otra, lo que permite copiar líneas que sean muy parecidas y modificarlas posteriormente.

El ensamblador

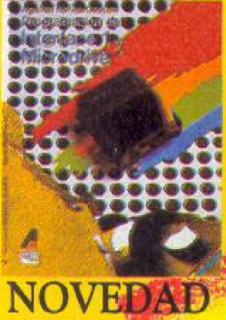
La tarea de convertir los mnemotécnicos (los nombres de las instrucciones) a unos y ceros comprensibles por el ordenador es realizada por esta parte del programa. Si hubiese que emplear un adjetivo para definirlo, sería «potente». Se ha planteado como un sistema destinado a profesionales e incluye muchas utilidades orientadas a la realización de programas largos.

Una de ellas son los Macros. Estos son un conjunto de instrucciones que se agrupan bajo un nombre y que cada vez que se referencia en el programa, se sustituyen por él realizando los cambios de variables apropiados. En cierto modo se pueden comparar a los procedimientos del Pascal, o a las palabras del Forth y del Logo.

También existe la posibilidad de realizar ensamblados parciales de partes del programa para unirlo todo posteriormente. Esto posibilita la realización de librerías de uso general que son posteriormente incorporadas a cada aplicación.

Una última posibilidad, quizás la más sorprendente y menos extendida, es la de realizar ensamblados desde cinta o microdrive. Cuando el programa es tan largo que no cabe en memoria para su ensamblado, el GENS lo graba en cinta con un formato especial que le permite irlo leyendo según se ensambla y así no ocupa espacio en memoria.

Como resumen se puede decir que este programa es uno de los más potentes, permitiendo la realización de aplicaciones profesionales con el Spectrum, pero como contrapartida está la dificultad de manejo.



NOVEDAD

PROGRAMACION DEL INTERFACE I Y MICRODRIVE

Agustín Núñez Castain

96 págs. 1.000 pts.

La extensión del Spectrum con el Interface I y el Microdrive supone un cambio drástico en la utilización de este microordenador. De ser un micro para videojuegos y aprendizaje pasa a ser un equipo con el que se pueden desarrollar aplicaciones serias, dentro del ámbito personal. Con imaginación, tu Spectrum y PROGRAMACION DEL INTERFACE I y MICRODRIVE puedes definir tu propio lenguaje y ampliar radicalmente las aplicaciones de tu microordenador.



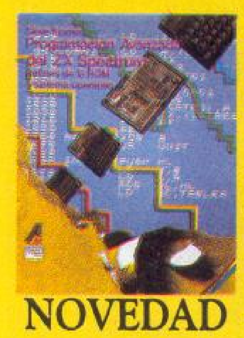
NOVEDAD

JUEGOS GRAFICOS DE AVENTURA PARA ZX SPECTRUM

Richard Hurley

128 págs. 1.300 pts.

Se trata de una recopilación de siete juegos de aventura, cada uno modelo de una técnica diferente de diseño y programación. Este es el primer libro dedicado específicamente al diseño de juegos de aventura gráficos, por lo que, no sólo interesa a los usuarios de ZX Spectrum, sino también a los usuarios de otros ordenadores con interés en el diseño de juegos.



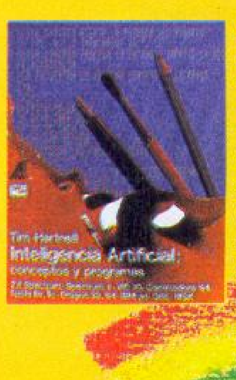
NOVEDAD

PROGRAMACION AVANZADA DEL ZX SPECTRUM: Rutinas de la ROM y Sistema Operativo

Steve Kramer

128 págs. 1.100 pts.

Es una obra fundamentalmente de referencia para consultar datos que necesitará cualquier programador en lenguaje máquina del Spectrum. El libro da toda la información sobre los programas que hay en la ROM del ZX Spectrum y cuál es el Sistema Operativo interno del microordenador. Explica cómo podemos utilizar la potencia interna de la máquina en nuestros programas. Su tratamiento esquemático y claro de forma, permite que la información se encuentre rápidamente.

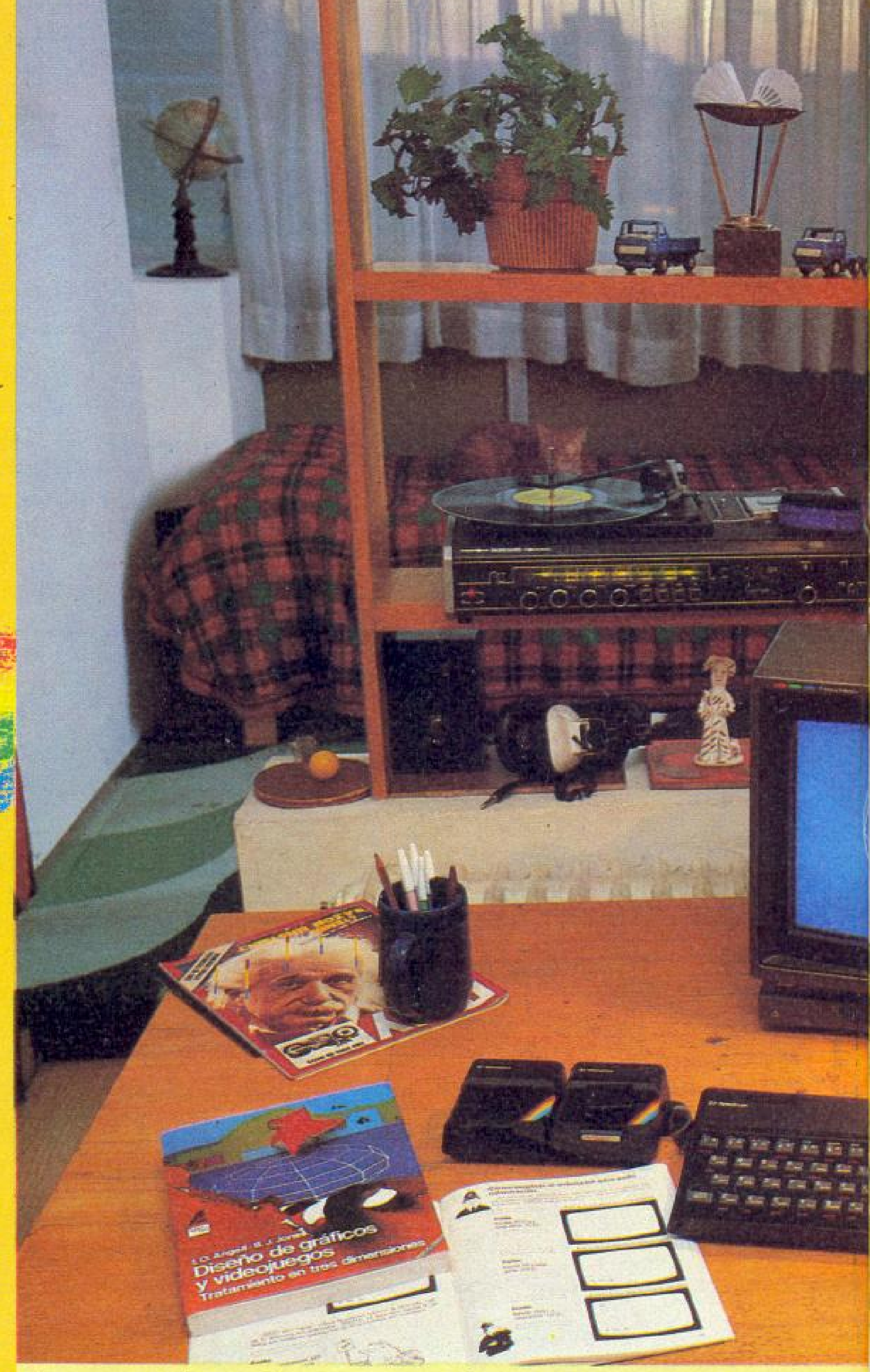


INTELIGENCIA ARTIFICIAL: CONCEPTOS Y PROGRAMAS

Tim Hartnell

272 págs. 1.300 pts.

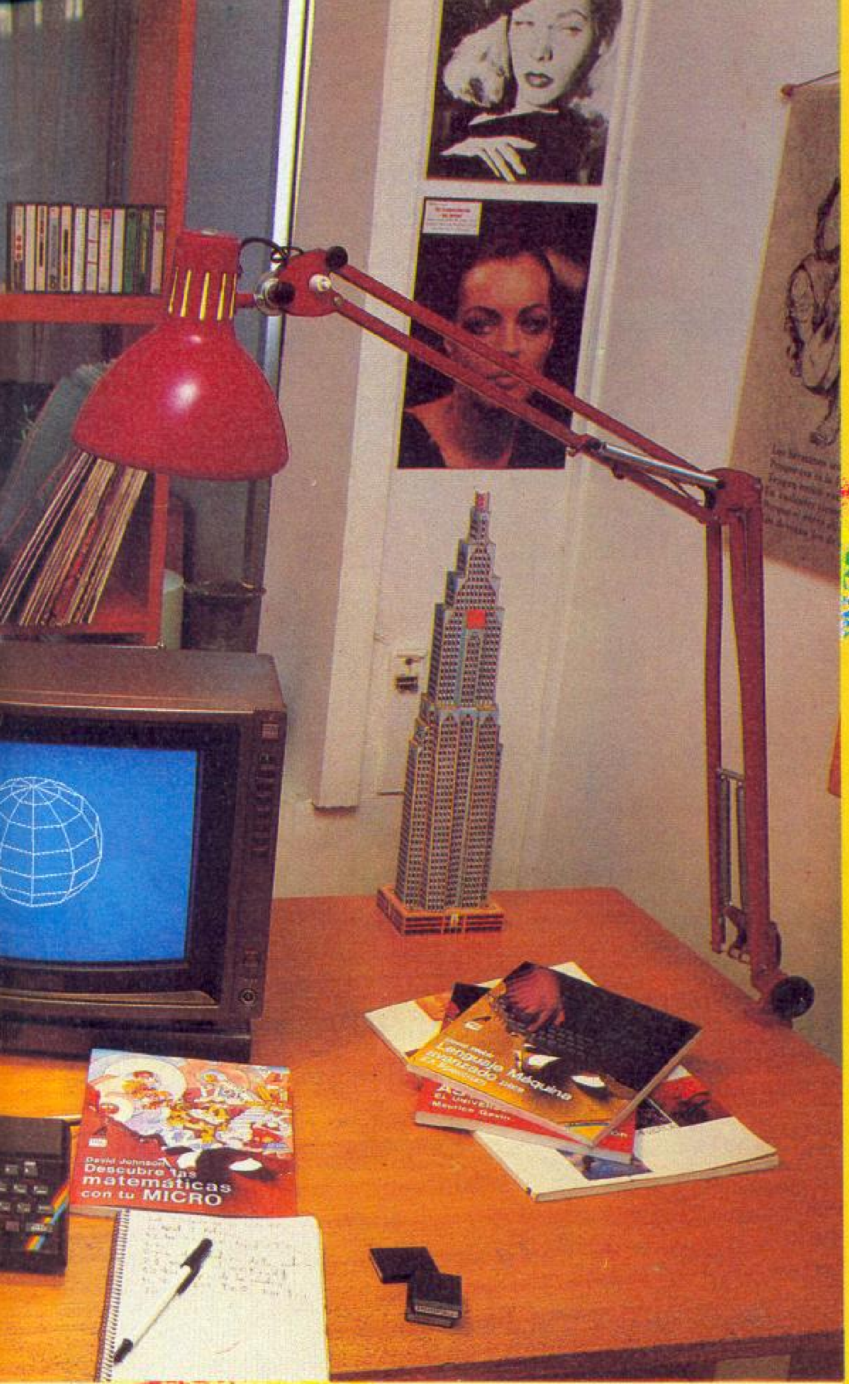
Una aventura fascinante en un reino donde se difuminan las barreras entre el hecho científico y la ciencia ficción vivida a través de algunos de los ejemplos más famosos: SILOGISMOS: Un programa que razona. TICTAC: Un juego de TRES EN RAYA que aprende a medida que juega. BLOQUELANDIA: Un programa que se comunica con nosotros en castellano. DOCTOR: La versión BASIC más sofisticada que se ha publicado nunca del famosísimo ELIZA. TRADUCTOR: Un programa que traduce el español al "spanglish". HANSHAN: Un poeta automático. APRENDESOLO: Un sistema experto que aprende.



Anaya Multimedia pone a tu disposición un camino segur

UNA BIB
MUY PE





para formar tu biblioteca informática.

LIOTECA RSONAL.

YA
TIMEDIA



VIFI ANAYA

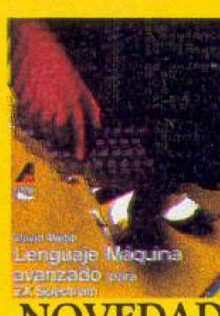
DISEÑO DE GRAFICOS Y VIDEOJUEGOS. TRATAMIENTO EN TRES DIMENSIONES.



Ian O. Angel y Brian J. Jones
(Incluye cassette)
392 págs. 2.900 pts.

Libro de carácter práctico con multitud de programas, que tiene que ser leído con el ordenador en funcionamiento. Trata sobre cómo utilizar un microordenador ZX Spectrum para dibujar figuras en dos y tres dimensiones y moverlas o manipularlas. Profundiza en los temas importantes de diseño de gráficos por ordenador y el último capítulo está dedicado al diseño de videojuegos.
El libro va acompañado de un cassette.

LENGUAJE MAQUINA AVANZADO PARA ZX SPECTRUM



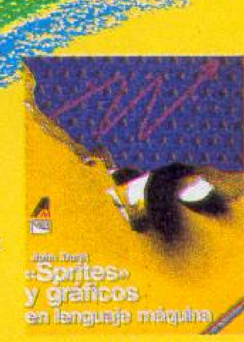
NOVEDAD

David Webb
192 págs. 1.300 pts.

De carácter muy práctico y con multitud de listados y rutinas de ejemplo, desarrolla técnicas avanzadas que utilizan los profesionales del diseño de videojuegos para conseguir efectos sofisticados en el ordenador.
Va dirigido a programadores del ZX Spectrum que deseen mejorar su conocimiento de las posibilidades del Z80 para diseñar programas rápidos.

OTROS TITULOS

"SPRITES" Y GRAFICOS EN LENGUAJE MAQUINA. (ZX SPECTRUM)



John Durst
184 págs. 1.350 pts.

EL LIBRO GIGANTE DE LOS JUEGOS PARA ZX SPECTRUM

Tim Hartnell
272 págs. 1.250 pts.

Adquiéralos en su librería habitual. Si no le es posible o desea que le enviemos nuestro catálogo, envíe este cupón a:
GRUPO DISTRIBUIDOR EDITORIAL
D. Ramón de la Cruz, 67
28001 MADRID

☐ Les ruego me envíen el catálogo de su editorial.
☐ Les ruego que me envíen los siguientes títulos:
....
....
TOTAL .... ptas.
☐ Adjunto talón bancario a GRUPO DISTRIBUIDOR EDITORIAL, S.A.
☐ Pagaré contrarreembolso (+ 125 pesetas de gastos de envío).
☐ Giro postal.
Nombre ....
Profesión ....
Dirección ....
C.P. .... Localidad ....
Provincia ....
TS

Programas

El BASIC letra a letra

El programa que nos envía Pedro Montolio no es un curso de BASIC, aunque el título que le hayamos puesto parezca indicarlo, sino que es una utilidad destinada a resolver un problema que le surge a muchos usuarios del Spectrum, sobre todo cuando ya han usado otros ordenador antes.

De todo el mundo es sabido que en todos los ordenadores excepto en los de Sir Clive las instrucciones BASIC se deben teclear letra a le-

```
64700 3E 00       ld a,0
64702 32 CE FD    ld (64974),a
64705 CD E8 FC    call 64744
64708 2A 59 5C    ld hl,(23641)
64711 7E          ld a,(hl)
64712 FE 0D       cp 13
64714 28 0B       jr z,64727
64716 E5          push hl
64717 CD C5 FD    call 64965
64720 DC B2 FD    call c,64946
64723 E1          pop hl
64724 23          inc hl
64725 18 F0       jr 64711
64727 3A CE FD    ld a,(64974)
64730 A7          and a
64731 28 10       jr z,64761
64733 21 BC FC    ld hl,64700
64736 E5          push hl
64737 FD 36 00 FF ld (iy+0),255
64741 C3 B4 12    jp 4788
64744 2A 59 5C    ld hl,(23641)

64747 7E          ld a,(hl)
64748 FE 0D       cp 13
64750 C8          ret z
64751 FE 0E       cp 14
64753 00          nop
64754 00          nop
64755 CC D1 FD    call z,64977
64758 23          inc hl
64759 18 F2       jr 64747
64761 FD CB 01 7E bit 7,(iy+1)
64765 20 07       jr nz,64774
64767 21 BC FC    ld hl,64700
64770 E5          push hl
64771 C3 B7 12    jp 4791
64774 CD 03 13    call 4867
64777 FD 36 00 FF ld (iy+0),255
64781 2A 59 5C    ld hl,(23641)
64784 CD A7 11    call 4519
64787 21 BC FC    ld hl,64700
64790 E5          push hl
64791 C3 B4 12    jp 4788

64794 11 96 00    ld de,150
64797 0E A4       ld c,164
64799 06 01       ld b,1
64801 2A CC FD    ld hl,(64972)
```

```
64804 0C          inc c
64805 79          ld a,c
64806 B7          or a
64807 C8          ret z
64808 1A          ld a,(de)
64809 FE 80       cp 128
64811 30 0B       jr nc,64824
64813 CD 47 FD    call 64839
64816 BE          cp (hl)
64817 20 0C       jr nz,64831
64819 23          inc hl
64820 13          inc de
64821 04          inc b
64822 18 F0       jr 64808
64824 D6 80       sub 128
64826 CD 47 FD    call 64839
64829 BE          cp (hl)

64830 C8          ret z
64831 1A          ld a,(de)
64832 13          inc de
64833 FE 80       cp 128
64835 38 FA       jr c,64831
64837 18 D8       jr 64799
64839 FE 41       cp 65
64841 D8          ret c
64842 FE 5B       cp 91
64844 D0          ret nc
64845 C6 20       add a,32
64847 C9          ret
64848 21 72 FD    ld hl,64882
64851 ED 5B CC FD ld de,(64972)
64855 46          ld b,(hl)
64856 C5          push bc
64857 23          inc hl
64858 0E 00       ld c,0
64860 1A          ld a,(de)
64861 96          sub (hl)
64862 B1          or c

64863 4F          ld c,a
64864 23          inc hl
64865 13          inc de
64866 10 F8       djnz 64860
64868 79          ld a,c
64869 C1          pop bc
64870 4E          ld c,(hl)
64871 B7          or a
```





tra en lugar de pulsar una sola que genere toda la palabra. Esto, en principio, parece un adelanto y no un problema, pero nos obliga a recordar la posición de cientos de instrucciones y cuando se trata de una no muy usada acabamos diciendo eso de «donde está esa» y nos tenemos que revisar todo el teclado, además en muchos casos resulta más fácil teclear letra a letra que no por medio de combinaciones de «SYMBOL SHIFT» y «CAPS SHIFT» que nos dan apariencia de pulpos. El programa en cuestión viene a paliar este problema por medio de una solución sencilla y bonita. Una pequeña rutina en código máquina que hace que podamos teclear los comandos como en cualquier otro ordenador.

Damos dos listados distintos del programa. El primero es un cargador hexadecimal hecho en BASIC que nos mete la rutina en memoria y la deja lista para funcionar. Las líneas 48 y 49 de este programa sirven para salvarlo y verificarlo, por lo que pueden suprimirse. Tener cuidado al introducirlo ya que no hay suma de comprobación y el

```
64872 C8              ret z
64873 23              inc hl
64874 3E 0D           ld a,13
64876 BE              cp (hl)
64877 06 00           ld b,0
64879 C8              ret z
64880 18 E1           jr 64851

64882 07 69 6E 76 65 72 73 65 DD
 05 69 6E 70 75 74 EE 03 69 6E 6
B D9 02 69 6E BF 03 61 74 6E B7
02 61 74 AC 0D
64917 CD 1A FD        call 64794
64920 79              ld a,c
64921 FE AC           cp 172
64923 28 B3           jr z,64848
64925 FE BF           cp 191
64927 28 AF           jr z,64848
64929 C9              ret
64930 2A CC FD        ld hl,(64972)
64933 71              ld (hl),c
64934 23              inc hl
64935 05              dec b
64936 E5              push hl
64937 C5              push bc
64938 CD 18 10        call 4120
64941 C1              pop bc
64942 E1              pop hl
64943 10 F7           djnz 64936
64945 C9              ret
64946 22 CC FD        ld (64972),hl
64949 CD 95 FD        call 64917
64952 79              ld a,c

64953 FE 00           cp 0
64955 C8              ret z
64956 3E 01           ld a,1
64958 32 CE FD        ld (64974),a
64961 CD A2 FD        call 64930
64964 C9              ret
64965 FE 7B           cp 123
64967 D0              ret nc
64968 FE 61           cp 97
64970 3F              ccf
64971 C9              ret

64972 00 00 00 00 00
64977 06 06           ld b,6
```

```
64979 CD A8 FD        call 64936
64982 2B              dec hl
64983 C9              ret

65023 1F FE           defw 65055
65025 3E FD           ld a,253
65027 ED 47           ld i,a
65029 2A 3D 5C        ld hl,(23613)
65032 36 BC           ld (hl),188
65034 23              inc hl
65035 36 FC           ld (hl),252
65037 ED 5E           im 2
65039 C9              ret
65040 2A 3D 5C        ld hl,(23613)
65043 36 03           ld (hl),3
65045 23              inc hl
65046 36 13           ld (hl),19
65048 3E 3F           ld a,63
65050 ED 47           ld i,a
65052 ED 56           im 1
65054 C9              ret
65055 F5              push af
65056 E5              push hl
65057 3E FE           ld a,254
65059 DB FE           in a,(254)
65061 1F              rra

65062 38 0F           jr c,65079
65064 3E BF           ld a,191
65066 DB FE           in a,(254)
65068 1F              rra
65069 38 08           jr c,65079
65071 2A 3D 5C        ld hl,(23613)
65074 36 BC           ld (hl),188
65076 23              inc hl
65077 36 FC           ld (hl),252
65079 FD CB 01 DE     set 3,(iy+1)
65083 E1              pop hl
65084 F1              pop af
65085 FF              rst 56
65086 ED 4D           reti
65088 00              nop
65089 00              nop
65090 00              nop
65091 00              nop
65092 00              nop
65093 00              nop
65094 00              nop
```

programa no detectará ningún dato mal tecleado.

El otro listado es el desensamblando de esta rutina que junto a los comentarios que incluimos a continuación proporcionará una interesante información a los amantes del código máquina.

Descripción de funcionamiento

El programa que se activa al hacer un RANDOMIZE USR 65025 consta de dos partes. La primera está situada entre las posiciones 65023 y 65039 y su función es desviar el sistema de interrupciones (para suprimir el modo K del cursor y comprobar si están pulsadas CAPS SHIFT y ENTER) y el sistema de errores, dirigiéndolo a la segunda rutina.

Se encuentra situada en 64700 y es la que compila y comprueba la sintaxis de las órdenes introducidas. Traduciendo los comandos introducidos a la sintaxis normal del Spectrum, saltando a continuación a la dirección 4788 para comprobar que no haya errores.

También está implementada la desconexión de estos procesos y la vuelta al modo normal de introducción por medio de un RANDOMIZE USR 65040.

El programa también se detiene después de un NEW por lo que se deberá repetir la orden RANDOMIZE USR 65025.

Por último indicar que después de un RUN o CLEAR, o con la parte inferior de la pantalla vacía hay que teclear simultáneamente CAPS SHIFT y ENTER.

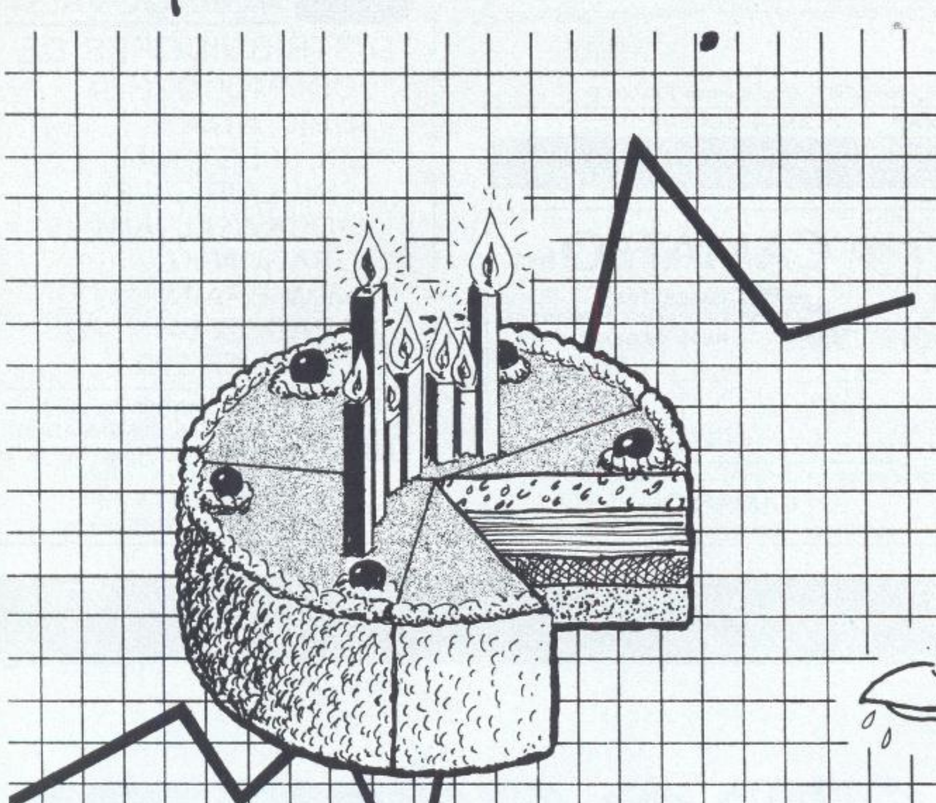
Pedro Montolio

```
   0>REM PMA
   2 CLEAR 49999
   3 RESTORE 0010: LET a=10: LET
 b=11: LET c=12: LET d=13: LET e
=14: LET f=15: LET pos=64700
   4 READ a$
   5 FOR m=0 TO 9
   6 POKE pos,16*VAL a$(2*m+1)+V
AL a$(2*m+2): LET pos=pos+1
   7 NEXT m
   8 IF pos=64990 THEN  LET pos=
65023
   9 IF pos<>65093 THEN  GO TO 0
004
  10 DATA "3e0032cefdcde8fc2a59"
  11 DATA "5c7efe0d280be5cdc5fd"
  12 DATA "dcb2fde12318f03acefd"
  13 DATA "a7281c21bcfce5fd3600"
  14 DATA "ffc3b4122a595c7efe0d"
  15 DATA "c8fe0e0000ccd1fd2318"
  16 DATA "f2fdcb017e200721bcfc"
  17 DATA "e5c3b712cd0313fd3600"
  18 DATA "ff2a595ccda71121bcfc"
  19 DATA "e5c3b4121196000ea406"
  20 DATA "012accfd0c79b7c81afe"
  21 DATA "80300bcd47fdbe200c23"
  22 DATA "130418f0d680cd47fdbe"
  23 DATA "c81a13fe8038fa18d8fe"
  24 DATA "41d8fe5bd0c620c92172"
  25 DATA "fded5bccfd46c5230e00"
  26 DATA "1a96b14f231310f879c1"
  27 DATA "4eb7c8233e0dbe0600c8"
  28 DATA "18e107696e7665727365"
  29 DATA "dd05696e707574ee0369"
  30 DATA "6e6bd902696ebf036174"
  31 DATA "6eb7026174ac0dcd1afd"
  32 DATA "79feac28b3febf28afc9"
  33 DATA "2accfd712305e5c5cd18"
  34 DATA "10c1e110f7c922ccfdcd"
  35 DATA "95fd79fe00c83e0132ce"
  36 DATA "fdcda2fdc9fe7bd0fe61"
  37 DATA "3fc9c67001000000606cd"
  38 DATA "a8fd2bc9000000000000"
  39 DATA "1ffe3efded472a3d5c36"
  40 DATA "bc2336fced5ec92a3d5c"
  41 DATA "36032336133e3fed47ed"
  42 DATA "56c9f5e53efedbfe1f38"
  43 DATA "0f3ebfdbfe1f38082a3d"
  44 DATA "5c36bc2336fcfdcb01de"
  45 DATA "e1f1ffed4d0000000000"
  46 CLS : PRINT "RECUERDE"'' "ON
:  RANDOMIZE USR 65025"''"OFF: RA
NDOMIZE USR  65040"''"ON2: CAPS S
HIFT+ENTER"
  47 STOP
  48 SAVE "compila": GO TO 48
  49 VERIFY "compila": GO TO 49
```

Guía del comprador de Todospectrum

Programas

Logosel



Con este curioso nombre se esconde un interesante programa escrito con la versión española del LOGO por Alfredo Lorente. Se trata de una rutina que dibuja gráficos de barras y de tarta en pantalla y que puede resultar de gran interés para usos estadísticos y de presentación gráfica de datos, además de enseñar más posibilidades de este lenguaje educativo.

Una importante particularidad del programa es que los datos se le introducen desde el teclado y el número de éstos puede ser variable según haga falta. Esto se hace aprovechando una estructura especial del LOGO, las listas, que permite hacer manejos no posibles con el BASIC.

```
?
PARA ENTRADADATOS
MTX
PONCURSOR [O 4]
ESCRIBE [Introduzca LOS DATOS EN
 FORMA DELISTA.]
ESCRIBE [DEJE UN ESPACIO ENTRE D
ATO__ Y_ DATO.]
ESCRIBE [CUANDO TERMINE PULSE EN
TER.]
ESCRIBE [Asegurese que los datos
 son los correctos antes de puls
ar ENTER]
PONCURSOR [O 12]
ASIGNA "lis LL
MTX
ESCRIBE [Si desea copia impresa
entre 's']
SI LCAR = "s [conimpresora]
FIN

PARA PRESENTACION
MTX
PONCURSOR [12 O]
ESCRIBE "LOGOSEL
ESCRIBE "
ESCRIBE [Este programa realizara
 un]
ESCRIBE [grafico de barras y otr
o - de]
ESCRIBE [tarta de los datos intr
oducidos.]
ESCRIBE "
FLASH
```

```
ESCRIBE [Para pasar de pagina pu
lse una - tecla]
NORMAL
PAUSA
FIN

PARA LOGOSEL
PRESENTACION
ENTRADADATOS
BARRAS
TARTA
FIN

PARA BARRAS
PRBAR
INICB
ASIGNA "e ESCA
ASIGNA "f 255 / CUENTA :lis
REC :lis
PONPOS [-128 -88]
ET
PAUSA
BP
FIN

PARA TARTA
PRTAR
ASIGNA "c 0
SUMAS :lis
ASIGNA "e 360 / :c
ASIGNA "d 0
CIR :lis
PAUSA
BP
FIN

PARA PRBAR
MTX
ESCRIBE [- - - Diagrama de barra
s - - -]
ESCRIBE "
ESCRIBE [Cada barra representa a
 un dato de izda a dcha en el or
den de entrada]
ESCRIBE "
FLASH
ESCRIBE [/ / / / PULSE UNA TECLA
 / / / /]
NORMAL
PAUSA
FIN

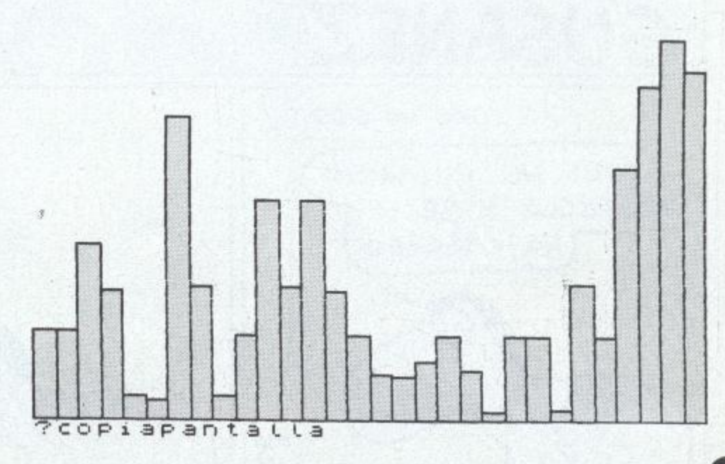
PARA INICB
LIMPIA
SINLAPIZ
PONPOS [-128 -88]
CONLAPIZ
FIN

PARA ESCA
DEVUELVE 175 / MAYOR
FIN

PARA REC :al
AV ( PRIMERO :al ) * :e
GD 90
AV :f
GD 90
AV ( PRIMERO :al ) * :e
GD 180
SI ( CUENTA :al ) = 1 [alto]
REC MENOSPRIMERO :al
FIN

PARA PAUSA
SI TECLA? [alto]
PAUSA
FIN

PARA PRTAR
MTX
ESCRIBE [- - - Diagrama de secto
res - - -]
```



Programas

58

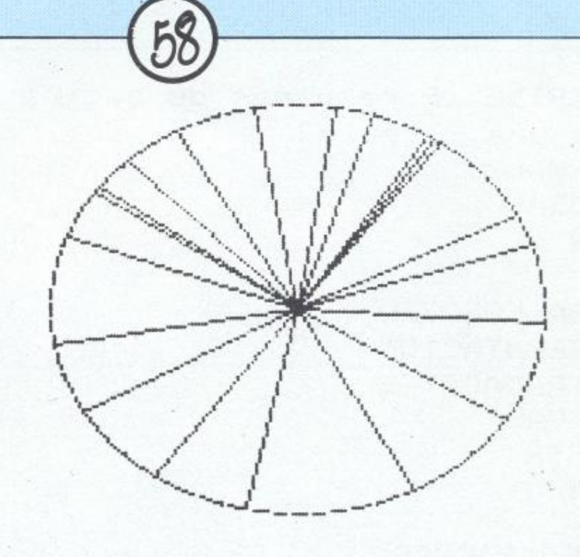
```
ESCRIBE [Cada sector representa
a un dato en el orden de entrada
]
ESCRIBE "
FLASH
ESCRIBE [/ / / / PULSE UNA TECLA
 / / / /]
NORMAL
PAUSA
FIN

PARA SUMAS :las
ASIGNA "c :c + PRIMERO :las
SI ( CUENTA :las ) = 1 [alto]
SUMAS MENOSPRIMERO :las
FIN

PARA CIR :las
CIRCIA :d 70 ( PRIMERO :las ) *
:e
SI ( CUENTA :las ) = 1 [alto] [c
ir menosprimero :las]
FIN

PARA BUS :b
SI ( ELEMENTO :b :lis ) > ( :a )
 [asigna "a elemento :b :lis]
SI :b = CUENTA :lis [alto]
BUS :b + 1
FIN

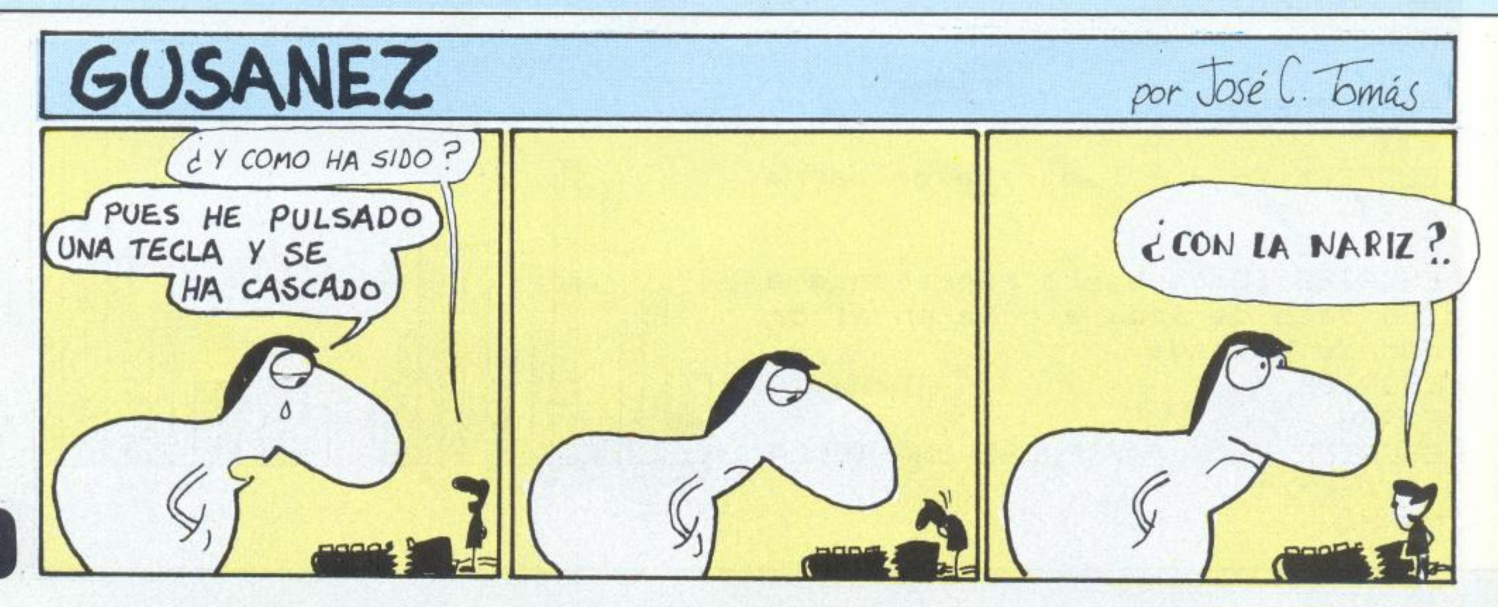
PARA MAYOR
ASIGNA "a ELEMENTO 1 :lis
```



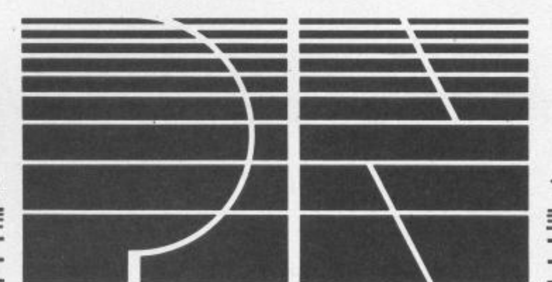
```
BUS 2
DEVUELVE :a
FIN

PARA CIRCIA :a :ra :nu
SINLAPIZ
AV :ra - 0.1
CONLAPIZ
AV 0.1
SI :a > ( :nu + :d ) [sinlapiz c
entro gi :a conlapiz av :ra sinl
apiz re :ra asigna "d :a alto]
SINLAPIZ
CENTRO
GI :a
CIRCIA :a + 1 :ra :nu
FIN

?
```



Paseo de Gracia, 11 - Esc. C., 2.º 4.ª



Tel. (93) 318 24 53 - 08007 Barcelona

GESTION

S.I.T.I. V.3 * — **4.000**
Al comprar esta versión abonamos 3.000'–ptas por cualquier versión anterior.

Context V.7 * — **4.000**
Tratamiento de Textos.
Funciona con cualquier impresora.
Cassette y/o microdrive. 64 col. en pantalla e impresora.

Context V.8 * — **4.000**
Nueva versión. Acentos graves y agudos.
Copy en alta resolución. Versiones para Seikosha SP-800, SP-1000 y Riteman F+

Adaptador SITI-CONTEXT — **2.000**
Permite pasar información del SITI al CONTEXT.

M.D.S. –Sistema Operativo para Microdrive — **7.000**
Conjunto de nuevos comandos BASIC que permiten Acceso Aleatorio a Ficheros en Microdrive con un tiempo medio de acceso de 4 segundos.

CONTABILIDAD PIN* — **3.000**
Plan contable. 200 cuentas, 2000 asientos.
Hasta 9.000.000.000. Balance con activo-pasivo, cta. resultados. Utiliza el S.O.M.D.S.
Cualquier impresora 80 col.

Kit Utilidades Discovery 1 — **2.000**
10 utilidades CAT extendido. ON ERROR, Set de caracteres del Amstrad, etc.

***Disponible en disco para Discovery 1 al precio de 5.000'–ptas.**

AJUSTE DE CABEZALES CASSETTE	**2.500**
SINTETIZADOR DE VOZ	**3.000**
MULTI-COPYS (Copys desde 2 cm. hasta 70 cm.)	**3.000**
COPY GRISES (F+, SP-800, SP-1000, GP-550)	**2.500**
COPY RS-232	**2.500**
COPY SERIE RITEMAN F+	**2.500**



Fotografía digitalizada y pasada a impresora con el Copy de Grises.



HARDWARE

Discovery 1 + Kit utilidades	**55.000**
Discos 3 1/2"	**8.000**
Cable impresora Discovery	**3.500**
Interface monitor	**3.900**
Interface sonito TV	**3.500**
Interface Centronics	**8.000**
Lápiz óptico + Sistema de dibujo	**4.850**
Kit Teclado Spectrum +	**8.500**
Teclado Saga 1	**14.000**
Impresora Riteman F + (Centronics) *	**69.000**
Impresora S-P 1000 (Centronics) *	**69.750**
Monitor CIAEGI F. Verde	**24.000**
Monitor CIAEGI F. Ambar	**24.750**

NOVEDADES PIN

Alimentación Ininterrumpida — **9.750**
No se pierde la información por corte de luz o bajada de Tensión . 1'30 h. de autonomía.
Recarga automática.

Digitalizador de Imágenes P-1024 — **35.000**
Digitaliza cualquier imagen impresa y la introduce en el ordenador donde se puede tratar. (Tramar, mezclar, siluetear, etc.). Muy fácil de usar.

*** OFERTA ESPECIAL**
Impresora + Interface Centronics + Context V.8 + Copy Grises — **72.000**

TIENDA AL PUBLICO EN BARCELONA
PEDIDOS POR CORREO O TELEFONO
Envíos contra reembolso a toda España
200 ptas. gastos de envío
En tu domicilio en 3-4 dias

Enviar a: PIN, P.º de Gracia, 11, Esc. C, 2º4º –08007 Barcelona

Nombre ____________________
Dirección ____________________
Población ____________________
Pedido ____________________

Preguntas y respuestas

NOTA DE LA REDACCION

Por un error lamentable, la portada del número anterior se decía que estaba hecha por Puchecha cuando en realidad era de Juan Carlos Centeno, un lector que nos dio buena muestra de sus cualidades artísticas.

Desde estas páginas queremos aclarar el error y animar a otros lectores a enviarnos sus portadas para poder convertirse en auténticos portadistas, viendo reflejado su trabajo en nuestras páginas.

P **Aunque poseo mi Spectrum desde hace poco, me intereso por todo lo relacionado con lenguajes y utilidades, al contrario que mi hermano que sólo quiere jugar. Por eso leo todo lo que publican referente a estos temas y practico bastante, sobre todo con el Pascal aunque me cuesta mucho, ya que el manual que viene con el programa no es muy claro.**

Paramí es más teórica que preguntar su funcionamiento. Leyendo dicho manual, llegué a un párrafo en el que se decía qué era un compilador, mientras que el BASIC dice que es un intérprete. Se que ambos pasan de estas instrucciones a código máquina, pero no la diferencia entre ambos. ¿Podríais decirme la diferencia entre ambos?

Miguel González De Lara
Madrid

P *Como tú bien dices los intérpretes y compiladores de lenguajes traducen a código máquina las instrucciones de los lenguajes respectivos. La diferencia es cómo lo hacen. Un intérprete (como el BASIC) tiene guardado el programa original y lo traduce cuando lo va a ejecutar. En cambio un compilador lo traduce previamente y almacena el código máquina. Esto hace que los lenguajes interpretados sean más fáciles de modificar, al guardar el original y poder ejecutarlo inmediatamente. En cambio los compiladores (como el PASCAL) tienen que hacer la traducción previamente a la ejecución y si se quiere modificar el programa hay que volver a compilar. Como contrapartida los programas compilados al no necesitar la traducción, se ejecutan mucho más rápidamente que los interpretados.*

En general se puede decir que los lenguajes interpretados son más útiles para el aprendizaje y los compilados lo son para el desarrollo de programas profesionales. No obstante lo más interesante es trabajar con el que uno se sienta más a gusto.

R **Quisiera pediros una aclaración con respecto al programa «Pitufolandia», aparecido en el TODOSPECTRUM del mes de mayo. Una vez seguidos los pasos que se detallan en la información, pasado el programa a cinta, revisado y ejecutado, tengo que efectuar el RUN en cada uno de los listados, cuando en un principio creía que saldría directamente y las tres partes del programa se unirían en un todo. ¿Me he equivocado en algo o lo he hecho mal? Tengo que deciros que, aunque hago mis «pinitos», soy un poco novato en esto todavía. Espero vuestra respuesta.**

José M. González
Palma de Mallorca

P *El problema a que haces referencia se resolverá si vuelves a pasar a cinta cada una de las partes del programa utilizando SAVE «nombre» LINE 1. De esta forma cada vez que se cargue una de las partes, ésta se ejecutará automáticamente. Esperamos que disfrutes del juego que tan pacientemente has tecleado.*

R **He tecleado el programa publicado en la revista número 13 «Arana Soft» y desde luego el resultado es satisfactorio y muy útil para mis fines si pudiera solucionar el problema de relocalización. En efecto,**



al trabajar con impresora y tener que estar conectada al interface Centronics de Indescomp, el volcado de la EPROM coloca el RAMTOP en la dirección 64515, y como el programa en cuestión coloca RAMTOP en 60159 y ocupa 5157 bytes el resultado es que coinciden el final de uno con el principio del otro. Y ésta es mi consulta: ¿hay alguna forma de poder utilizar simultáneamente el programa y la impresora?, y si es así, ¿cuál es la forma de hacerlo?

Emilio Pineda
Madrid

R *El problema que planteas tiene solución aunque ésta no es sencilla, ya que se trata de relocalizar de nuevo todo el programa, para lo cual deben modificarse todas las llamadas y saltos absolutos que éste usa. Prometemos mandarte, en cuanto realicemos los cambios oportunos, copia del programa para que puedas disfrutar de él conjuntamente con tu impresora.*

P **Al intentar ejecutar el programa «Un nuevo operativo para el Spectrum», aparecido en el número 13 de su revista, me ha sido imposible hacerlo, ya que el ordenador se me quedaba «bloqueado» cada vez que pulsaba la tecla ENTER.**

El Spectrum que poseo es de 48 K transformado a «Plus» por Hissa, equipado con Interface **1 y** microdrives. **Adquirido en agosto de 1984, la transformación se hizo en abril pasado.**

Como en el artículo se decía que con el Interface **1 podían aparecer problemas, procedí a desconectarlo y cargué el programa en** cassette, **después de repetir el laborioso trabajo de carga, pero el problema continúa. Tras cargar el programa, éste se autoejecuta, carga los** bytes **y aparece en caracteres comprimidos al mensaje «© ARANA SOFTWARE». Pero al introducir un comando y pulsar ENTER el ordenador queda «colgado» y no ejecuta el comando, tanto si trabajas en modo directo como si listas una instrucción. He repasado varias veces las DATA por si había algún error no detectado por el programa cargador, pero no encuentro ninguno.**

Por lo expuesto les agradecería que me indicasen qué es lo que no funciona y, en su caso, las correcciones que debo hacer para que el programa funcione correctamente. No creo que el ordenador esté averiado, pues con cualquier otro programa funciona correctamente. En espera de sus noticias se despide atentamente.

Francesc Ibarz
Reus (Tarragona)

R *Según las comunicaciones de otros lectores, el programa no ha dado ningún problema del tipo que tú mencionas, con lo que quedan descartados errores de imprenta o del programa en sí. Por otra parte, el hecho de que poseas* Interface *1 tampoco debería llevar a consecuencias tan drásticas como las que te han impulsado a escribirnos. Como tú mismo mencionas el problema no debe ser del ordenador ni del hecho de que haya sido transformado a Plus.*

Desde luego el programa cargador el largo y complejo como su equivalente en ensamblador, por lo que no es difícil que el error fuera tuyo a la hora de introducirlo. Te rogaríamos que revisaras cuidadosamente los datos introducidos y repasaras una vez más las instrucciones del programa. En último caso puedes mandarnos copia del mismo cassette *que nos encargaremos de revisar.*

Les agradecería que me indicasen:

1) Las principales rutinas de la ROM y sus direcciones de memoria.

El corcho

Intercambio programas para el Spectrum 16/48K. Interesados escribir a: Salvador Belenguer Carsi. Enrique Navarro, 7, pta. 10. 46020 Valencia.

VENDO SPECTRUM 48K EN PERFECTO ESTADO DE FUNCIONAMIENTO CON CABLES Y TRANSFORMADOR POR 30.000 PTS. REGALO CINTA HORIZONTES, MANUALES EN CASTELLANO E INGLES, MUCHOS PROGRAMAS COMERCIALES, DE APLICACIONES,... Y REVISTAS DE INFORMATICA. SI ERES DE BARCELONA TE ENSEÑO A USARLO. TAMBIEN LO CAMBIARIA POR TV COLOR 14 O 16 PULGADAS. CONTACTAR CON DANIEL. TEL.: 220 55 16. BARCELONA.

Cambio y vendo juegos para Spectrum 16/48K. Preguntar por Begoña. Tel.: 411 10 08. Bilbao.

Vendo consola de videojuegos completamente nueva con dos cartuchos de 10 y 6 juegos, respectivamente, y transformador por 9.000 pts. (negociables). También cambio programas para el ZX-Spectrum. Miguel Gómez Expósito. Marqués de la Hermida, 14-6.º J. 39090 Santander. Tel.: (942) 22 78 19 (por la tarde).

Intercambio programas ZX Spectrum 16/48K. Escribir a: Luis García Procopio. Urb. El Retortillo, p 5-2.º A. 11001 Chiclana de la Frontera (Cádiz). Mandar lista de programas.

Su anuncio puede ir aquí. Escríbanos a «El corcho». TODOSPECTRUM. Bravo Murillo, 377. 5.º A. 28020 Madrid.

Preguntas y respuestas

2) ¿Cómo puedo crear números aleatorios en código máquina?

3) ¿Cuál es la instrucción o rutina equivalente en código máquina a la instrucción BASIC PRINT PEEK «dirección»?

Juan R. Querol
Madrid

R *La ROM tiene docenas de rutinas, y en el limitado espacio de esta sección no podemos dedicarnos a este extenso tema, pero si estás interesado en el sistema operativo de tu Spectrum te recomendamos el libro de Steve Kramer Programación avazanda del ZX Spectrum,* en donde hallarás todas y cada una de las rutinas con indicaciones sobre su funcionamiento.

En cuanto a la creación de números aleatorios en código máquina puedes acudir al número de septiembre, donde, en esta misma sección, se le aclaraba a Miguel Angel Olivar lo referente a ese tema.

La forma más sencilla de solucionar tu tercera cuestión quizá sea la que utiliza la siguiente rutina:

```
PUSH HL
LD A,2
CALL 1601H
POP HL
LD A,(HL)
CALL 2D28H
JP 2DE3H
```

El par de registros HL deberá tener la dirección de memoria que pretendemos sea «peekeada». Esta rutina puede ubicarse en cualquier rincón de la memoria, y su funcionamiento es sencillo:

En primer lugar se salva en la pila el contenido del par HL que será modificado en la siguiente llamada. A continuación se procede a abrir el canal 2 con el fin de que lo que se imprima posteriormente lo sea en la parte principal de la pantalla; para ello basta dar al registro A el valor del canal a abrir y hacer una llamada a la subrutina de la ROM CHAN-OPEN (1601H).

Seguidamente, y tras recuperar en HL su valor original, damos al registro A el valor de la dirección señalada por HL y llamamos a una nueva subrutina de la ROM: STACK-A que almacena el valor de A en la pila del calculador. Tras esto basta salir por PRINT-FP, que se encargará de imprimir este valor y retornar.

P **Tengo mi ordenador Spectrum desde hace dos años y me funciona perfectamente y sin ningún tipo de problemas. Lo único que echo en falta es más velocidad en los programas de BASIC, ya que si no fuese por ese problema se podrían hacer juegos tan buenos como los que se venden comercialmente.**

Pensando en una solución para arreglar esto me enteré de que existían varios modelos del microprocesador Z-80 y que unos eran más rápidos que otros. Si se cambiase el que trae de fábrica por uno más rápido, los programas también irían más rápidos. ¿Es así? ¿Se produciría algún daño al hacer el cambio? Gracias anticipadas por su colaboración.

Francisco José López
Madrid

R *Como tú dices, existen varias versiones del Z-80. La primera iba como máximo a 2.5 megahercios (ciclos del reloj por segundo). La segunda, que es la que utiliza el Spectrum, se llama Z-80 A y va a 4 megahercios también como máximo. Posteriormente a ésta salieron otras dos: la Z-80 B a 6 megahercios y la Z-80 H a 10 megahercios.*

Pero estas velocidades de operación son, como hemos indicado, máximas y muchas veces va a una inferior, ya que esto no lo fija el microprocesador sino una electrónica asociada dentro del ordenador. Esta electrónica es la que fija la velocidad. Si a un microprocesador preparado para ir a más velocidad (el Z-80 H por ejemplo) se le hace funcionar a menos, no pasa nada especial y el ordenador seguirá funcionando igual.

DIRECTOR: Simeón Cruz
COORDINADOR EDITORIAL: Emiliano Juárez
REDACCION: Enrique Larreta
Juan Arencibia, Fernando García, José C. Tomás, Luis M. Brugarolas, Ricardo García, Santiago Gala
DISEÑO: Ricardo Segura

Editado por PUBLINFORMATICA, S. A.
Presidente: Fernando Bolín

Administración: INFODIS, S. A.
Gerente de Circulación y ventas: Luis Carrero
Producción: Miguel Onieva
Director de Marketing: Antonio González
Servicio al cliente: Julia González. Tel. 733 79 69
Administración: Miguel Atance
Jefe de Publicidad María José Martín
Dirección y redacción: Bravo Murillo, 377-5.º A. Tel. 733 74 13
Telex: 48877 OPZX e 28020 Madrid
Administración y Publicidad: Bravo Murillo, 377-3 E. Tels. 733 96 62/96
Publicidad Madrid: María Jose Martín
Publicidad Barcelona: María del Carmen Ríos, Olga Martorell. Pelayo, 12. Tel. (93) 318 02 89. 08001 Barcelona.
Depósito legal: M-29041-1984
Distribuye S.G.E.L. Avda. Valdelaparra, s/n. Alcobendas-Madrid.
Fotomecánica: Karmat, C/ Pantoja, 10. Madrid.
Fotocomposición: Artecomp.
Imprime: Héroes, C/ Torrelara, 8. Madrid.
Distribuidor en VENEZUELA,
SIPAM, S.A.
AVD. REPUBLICA DOMINICANA, EDIF. FELTREC – OFICINA 4B BOLEITA SUR CARACAS (VENEZUELA)
Esta publicación es miembro de la Asociación de Revistas de Información ari asociada a la Federación Internacional de Prensa Periódica, FIPP.

SUSCRIPCIONES:
Rogamos dirijan toda la correspondencia relacionada con suscripciones a:
TODOSPECTRUM
EDISA: Tel. 415 97 12
C/ López de Hoyos, 141-5.º
28002 MADRID
(Para todos los pagos reseñar solamente TODOSPECTRUM)
Para la compra de ejemplares atrasados diríjanse a la propia editorial
TODOSPECTRUM
C/ Bravo Murillo, 377-5.º A
Tel. 733 74 13-28020 MADRID

Si deseas colaborar en TODOSPECTRUM remite tus artículos o programas a Bravo Murillo 377, 5.º A. 28020 Madrid. Los programas deberán estar grabados en cassette y los artículos mecanografiados.
A efectos de remuneración, se analiza cada colaboración aisladamente, estudiando su complejidad y calidad.